MICROPROCESSORI

MICROPROCESSORI

GRUPPO EDITORIALE
JACKSON

INDICE

Capitolo 1

5 Introduzione

Capitolo 2

9 Cos'è un microprocessore

Capitolo 3

21 Tecnologie costruttive

Capitolo 4

35 Microprocessori a 8 bit più comuni (I)

Capitolo 5

85 Microprocessori a 8 bit più comuni (II)

Capitolo 6

121 Microprocessori a 8 bit più comuni (III)

Capitolo 7

163 Microprocessori a 16 bit

Capitolo 8

201 Conclusioni

CAPITOLO 1

INTRODUZIONE

nche se il tempo trascorso dalla comparsa dei microprocessori è relativamente breve, il loro impatto è stato notevole, e le persone attente alle innovazioni tecnologiche hanno compreso la rivoluzione apportata nel mondo dell'elettronica.

In effetti, i microprocessori aprono nuove strade e semplificano il progetto di sistemi complessi, in quanto dispongono di una notevole flessibilità e rendono possibile l'introduzione di nuove funzioni nei sistemi già esistenti.

Tra i molti vantaggi, è rilevante la diminuzione dei costi per innumerevoli applicazioni, nelle quali decine di circuiti integrati possono essere sostituiti da qualche *chip*. Il ridotto cablaggio e la miniaturizzazione del circuito consentono una maggior affidabilità, un minor consumo di potenza e, inoltre, una elevata facilità di diagnosi dei guasti, di riparazione, e di manutenzione degli apparecchi.I microprocessori hanno una notevole influenza sulla concezione del progetto, sul tempo e sui costi di progettazione di nuovi apparecchi e, di conseguenza, sui costi di fabbricazione.

I dispositivi a microprocessore sono utilizzati in numerose applicazioni, che comprendono un'ampia varietà di apparecchi, dalle comunicazioni all'industria, ai prodotti di consumo, agli strumenti di misura ed agli elaboratori. In tale gamma si possono citare sistemi svariati, come il controllo di processo, il controllo numerico, i terminali intelligenti, i giochi elettronici, gli strumenti autocalibranti, le apparecchiature biomediche, le applicazioni militari, il controllo di veicoli, l'elaborazione di dati, gli elettrodomestici, ecc.

Nello sviluppo di sistemi basati sui microprocessori intervengono tre fasi molto complesse: la fase di progetto, l'uso di un adeguato supporto software per lo sviluppo, e, infine, le tecniche per la creazione del software applicativo.

LSI: tecnologia che ha permesso di produrre i microprocessori

Tutti conoscono il transistore, un componente molto piccolo, ma visibile; si immagini però lo spazio che occuperebbero cinquantamila transistori.

Fortunatamente, negli ultimi anni, la tecnologia ha reso possibile la realizzazione di dispositivi molto complessi, con dimensioni molto ridotte.

Questi dispositivi, che per il colore e la forma hanno ricevuto il soprannome di *ragnetti*, sono i circuiti integrati (CI), costituiti da pastiglie, sia in plastica che in ceramica, contenenti una piccola piastra di silicio, materiale semiconduttore, con cui si costruiscono i transistori. Il vantaggio dei C.I. sta nel fatto che, invece di utilizzare dei transistori sciolti e collegarli esternamente, sfruttano una piastrina che contiene tutti i transistori e i relativi collegamenti.

Nel progetto dei circuiti integrati, delle apparecchiature speciali convertono gli schemi logici, rappresentati su carta con elementi AND, OR e NOT, negli equivalenti schemi elettronici a transistori, e questi ultimi in tracciati adatti che permettono di costruire il chip di silicio mediante un processo fotolitografico. Il vantaggio è enorme, dato che lo spazio occupato dagli elementi attivi si riduce, fotograficamente, a dimensioni dell'ordine dei micron, rendendo possibile l'integrazione di migliaia di funzioni logiche.

Per porte semplici, si usa l'integrazione a bassa scala (SSI = Small Scale Integration), dato che i circuiti da costruire sul chip non sono molti. In un personal computer, le porte logiche sono solitamente circuiti integrati a 14 piedini che si trovano sulla scheda principale.

Per funzioni più complicate si utilizza una integrazione a media scala (MSI).

Infine gli integrati più complessi, grandi e costosi, come i microprocessori, che costituiscono la CPU di un personal computer, si costruiscono con tecniche di integrazione a larga scala (LSI e VLSI).

Un fatto che occorre tenere presente è che le dimensioni del contenitore non dipendono tanto dalla complessità del chip, quanto dal numero di funzioni che occorre fornirgli dall'esterno, in ingresso e uscita.

Un microprocessore è solitamente inserito in un contenitore con un minimo di 40 piedini, poichè il numero di segnali che deve controllare è molto elevato. In realtà, il circuito integrato, normalmente, non occupa una superficie maggiore di 25 mm^2.

Hardware di un personal computer

La scheda principale, a volte l'unica esistente e necessaria per il completo funzionamento del computer, presenta, di solito, una disposizione dei com-

Foto 1.-Aspetto della scheda madre di un personal computer basato su un microprocessore, cuore dell'apparecchio.

ponenti che indica la diversità delle funzioni dei chips utilizzati. Prima di tutto, il *capofamiglia,* il microprocessore, sarà sempre facilmente identificabile: è un circuito integrato grande, con un contenitore solitamente a 40 piedini, come nel caso del 6502 (microprocessore dell'Apple), o dell'8088 (microprocessore dell'IBM-PC). Negli apparecchi più moderni, come il *Mac* Apple, il µP è inserito in un contenitore a 64 piedini: si tratta del 68000, molto più potente degli altri due, che, di conseguenza, necessita di più terminali.

Più avanti si vedrà come è fatto e come funziona un microprocessore.

Per ora, basta ricordare che il microprocessore viene definito anche Unità Centrale di Processo (o CPU), poichè il suo compito è di dirigere l'esecuzione dei programmi e il processo dei dati inseriti dall'utente.

Attorno alla CPU si trovano altri circuiti integrati che risultano normalmente più piccoli. Una buona parte di essi costituiscono la memoria, che funziona, relativamente alla CPU, come una specie di magazzino per i dati e gli ordini in corso di esecuzione. Infine, altri circuiti, noti come *integrati di ingresso/uscita,* permettono il collegamento della scheda principale con altre schede esterne, e il collegamento di quelli citati precedentemente tra di loro. Sulla scheda esiste un vasta gamma di altri integrati più piccoli, la cui funzio-

ne è di adattare i segnali che fluiscono tra i circuiti integrati principali, facilitandone il collegamento.

La necessità di questi piccoli circuiti ausiliari deriva dal fatto che, sia il microprocessore, sia la memoria, che i dispositivi di ingresso/uscita, sono progettati in modo che possano formare configurazioni molto diverse per vari scopi.

Questa versatilità obbliga ad aggiungere, per ogni tipo di configurazione, un certo numero di tali componenti ausiliari.

CAPITOLO 2

COS'E' UN MICROPROCESSORE

n questo capitolo verrà definito il microprocessore e ne verranno descritte le caratteristiche principali.

Definizione di microprocessore

Il microprocessore è un circuito integrato capace di eseguire un programma e di controllare i dispositivi necessari a tale esecuzione. Le principali applicazioni del microprocessore sono:

- la sostituzione di circuiti logici, utilizzabili per eseguire un unico compito, con circuiti programmabili in grado di risolvere diversi problemi con programmi distinti.
- il funzionamento come unità centrale di processo di un microelaboratore.

Caratteristiche di un microprocessore

I microelaboratori hanno caratteristiche strettamente dipendenti dal microprocessore su cui si basano, poichè sia la loro potenza, sia le altre prestazioni, sono condizionate dalle caratteristiche della loro CPU, costituita da un microprocessore. Le prestazioni caratteristiche di un microprocessore sono le seguenti:

- lunghezza della parola processata: le lunghezze più comuni della

parola nei microprocessori attuali sono 8 o 16 bit, anche se ne esistono alcuni che operano con parole di 4 o 32 bit. La precisione di calcolo del microprocessore e la sua capacità di indirizzamento aumenta con l'aumentare della lunghezza della parola elaborata;

- capacità di memoria: questa caratteristica è normalmente correlata con la lunghezza della parola elaborata. La capacità massima della memoria utilizzabile da un microprocessore è stabilita dalle sue capacità di indirizzamento. Tuttavia, microprocessori con parole della stessa lunghezza possono avere memorie diverse nella configurazione iniziale;
- velocità di esecuzione delle istruzioni: si definisce ciclo di istruzione, il tempo impiegato dal microprocessore per eseguire completamente un'istruzione. Tale caratteristica determina la velocità di esecuzione di un microprocessore. Il fattore da considerare nell'adottare questa misura come dato caratteristico, è che il ciclo è diverso in funzione del tipo di istruzione eseguita. Un'altra misura più omogenea è il ciclo macchina, cioè il tempo impiegato dal microprocessore per eseguire un'operazione elementare, che costituisce parte di una qualsiasi istruzione completa;
- registri speciali: un'altra caratteristica importante dei microprocessori è il numero dei registri speciali che possiedono. La maggioranza dispone di un unico accumulatore nell'unità aritmetico-logica, ma esistono microprocessori che incorporano due accumulatori, fatto che amplia la loro potenza e velocità di elaborazione. Analogamente, esistono due tendenze per gli altri registri interni: la prima consiste nell'utilizzare parte della memoria RAM come registri propri del microprocessore, la seconda è di inserire vari registri di lavoro all'interno del microprocessore;
- capacità di interrupt: l'esecuzione di un programma può essere interrotta in alcuni casi. Una caratteristica fondamentale del microprocessore è la capacità di recepire e gestire un determinato numero di interruzioni. Mediante tali interruzioni si possono stabilire le comunicazioni necessarie, sia con l'utente che con altri dispositivi del microelaboratore, senza che ciò influisca sulla corretta esecuzione del programma i corso;
- famiglia di circuiti complementari: la necessità di completare l'operatività del microprocessore, esige l'impiego di una serie di circuiti integrati adattabili allo stesso. In questo modo nascono

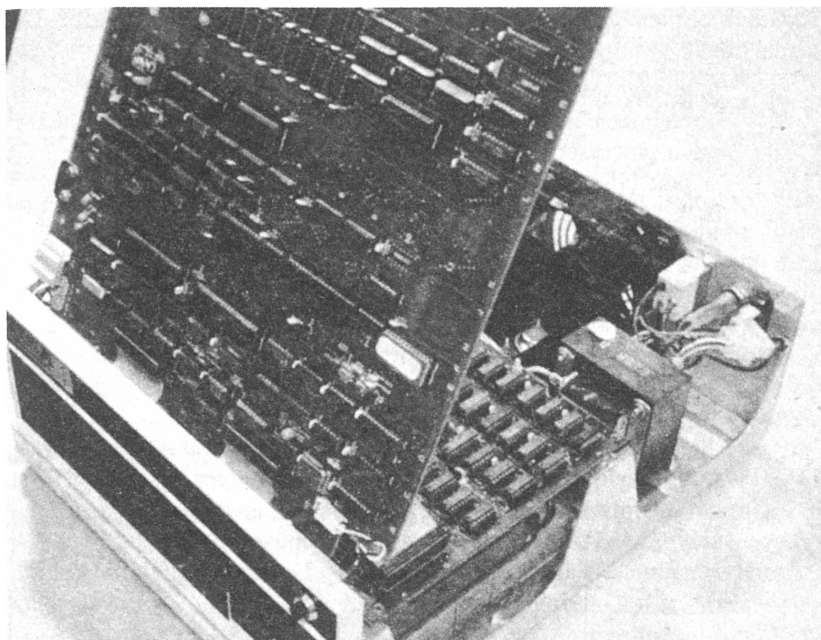



Foto 1.-Nonostante il volume minimo, il microprocessore funziona come cervello di potenti microelaboratori. Ad esempio, il microprocessore Z80 costituisce la CPU del sistema multiutente ALTOS-8010 che è visibile nella fotografia.

diverse *famiglie* di circuiti complementari. Ad esempio, si può citare la famiglia del 6800 o dell'8080, riferendosi non solo ai microprocessori MOTOROLA-6800 o INTEL- 8080, ma anche ai circuiti a questi adattabili.

Programmazione di un microprocessore

Quanto precedentemente esposto sui microprocessori porta a concludere che, a livello fisico (hardware), hanno la stessa configurazione dell'unità centrale di processo di un elaboratore. Questo vale non solo per la parte fisica, ma anche dal punto di vista logico (software), poichè il loro funzionamento è analogo. Il microprocessore funziona direttamente in linguaggio macchina, tuttavia, disponendo del corrispondente traduttore, si possono uti-

lizzare determinati linguaggi di programmazione più evoluti. La classificazione dei linguaggi utilizzati è la seguente:

- linguaggio macchina: le istruzioni del programma si scrivono direttamente in binario e sono immediatamente interpretabili dal microprocessore. In alcuni casi, invece di un linguaggio binario si può utilizzare una codificazione ottale o esadecimale. Un programma codificato direttamente in linguaggio macchina viene definito programma oggetto;
- linguaggi simbolici: invece di ricorrere a codificazioni binarie, come nel caso del linguaggio macchina, si può programmare con istruzioni simboliche. Questo tipo di programma viene definito sorgente, e deve essere tradotto in programma oggetto da un compilatore;
- linguaggi ad alto livello: rappresentano un passo avanti verso l'utilizzazione di linguaggi simili a quelli convenzionali e facili da interpretare dall'utente. Allo stesso modo dei linguaggi simbolici, i programmi redatti in linguaggio ad alto livello vengono definiti sorgente e, prima di essere eseguiti, devono essere tradotti in linguaggio macchina.

L'informazione nel microprocessore

Un microprocessore elabora due tipi di informazioni: quelle che verranno interpretate dall'unità di controllo, e i dati, di cui si occupa l'unità aritmetico-logica in accordo con quanto indicato nelle istruzioni. Si analizzerà, nei paragrafi successivi, il flusso di informazioni che si stabilisce nel microprocessore, quando questo esegue le tre istruzioni fondamentali di caricamento, memorizzazione, e biforcazione.

Esecuzione sequenziale

Le istruzioni che devono essere eseguite dal microprocessore sono costituite da due parti: codice di operazione e operando (uno o più). Ad esempio, nell'operazione di somma di due numeri (A + B), A e B sono gli operandi (in questo caso gli operandi coincidono con i dati reali) e il segno + il codice dell'operazione. L'elaborazione di ciascuna delle parti che costituiscono l'istruzione è svolta da diverse unità interne del microprocessore. I codici dell'operazione sono elaborati dall'unità di controllo, e gli operandi dall'uni-

tà aritmetico- logica.

Per inviare in modo sequenziale le parole binarie dell'informazione verso l'unità di controllo, o verso l'unità aritmetico-logica, ed ottenere la corretta esecuzione del programma, è necessario conoscere perfettamente le posizioni della memoria in cui risiede tale informazione. Le diverse posizioni della memoria sono identificate da uno specifico indirizzo. Per accedere in qualsiasi momento alla posizione di memoria necessaria, il microprocessore conta, mediante un registro specializzato per questo compito, definito *contatore di programma* o *contatore di indirizzi*. In ogni istante, il contenuto di tale registro coincide con l'indirizzo corrispondente alla posizione della memoria alla quale accederà il microprocessore, sia per leggere che per memorizzare in essa una informazione. Per il momento interessa conoscere il metodo seguito dal microprocessore per eseguire un programma, o un insieme di istruzioni costituite da codici di operazione (ordini) e operandi (dati o indirizzi corrispondenti alla posizione nella memoria dei dati che interessano).

Il processo di esecuzione inizia con la lettura da parte del microprocessore, del contenuto di una locazione di memoria, che, per quanto visto, è indicata dal contatore di programma. Perchè l'esecuzione avvenga automaticamente da questo istante, è necessario che la prima parola letta nella memoria, che può essere sia un'istruzione che un dato, coincida con un *codice di operazione,* in quanto, essendo un comando, indicherà al microprocessore qual'è la natura (codice di operazione o dato) della parola che sarà letta successivamente.

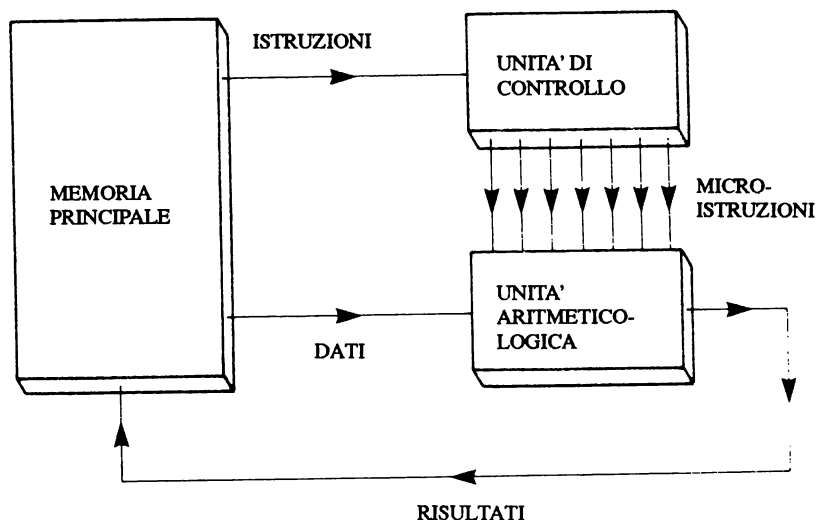
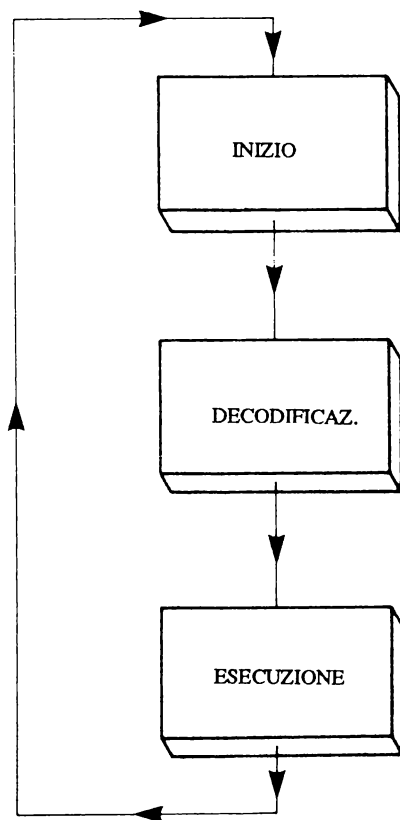



Fig. 1.-a) Il grafico indica il flusso dell'informazione all'interno di un sistema basato su microprocessore. b) Le tre fasi che costituiscono il trattamento dell'istruzione da parte del microprocessore

Questa prima parola dell'informazione che si legge nella memoria, raggiunge il microprocessore attraverso il *bus dei dati* e viene inviata ad un registro interno detto *registro delle istruzioni.* Fatto ciò, l'unità di controllo interpreta tale parola che, com'è stato detto, coincide con un codice operativo o ordine da eseguire. Dopo averla interpretata, l'unità di controllo conoscerà l'operazione da eseguire, e la natura della parola che deve leggere nella memoria. Se la nuova parola è un codice di operazione, verrà interpretata analogamente alla precedente; se si tratta invece di un operando, lo invierà all'unità aritmetico-logica e genererà gli ordini necessari perchè venga elaborato adeguatamente.

Il flusso delle informazioni si stabilisce attraverso tre vie:

- il bus dei dati, che ha il compito di leggere e memorizzare le informazioni nella memoria;
- il bus degli indirizzi, che indica la posizione in memoria dalla quale leggere o nella quale scrivere la parola dell'informazione;
- il bus di controllo, che trasmette gli ordini (microistruzioni) generati in seguito all'interpretazione dei codici operativi delle istruzioni.

Esecuzione di un'istruzione tipica

Il processo di esecuzione inizia con l'invio, al bus degli indirizzi, del contenuto del registro contatore degli indirizzi, in modo da puntare alla posizione di memoria che contiene la prima parte dell'istruzione: il codice operativo.

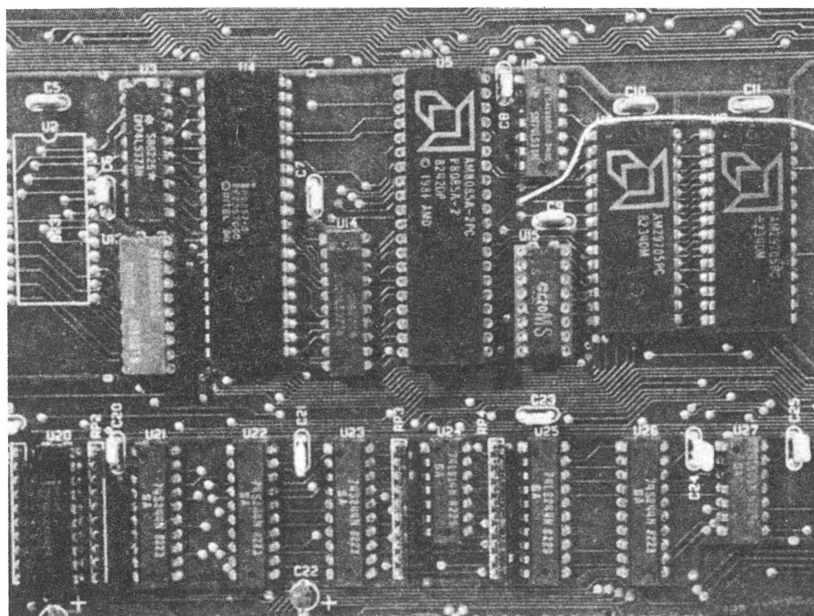



Foto 2.-Il microprocessore non è solo in grado di elaborare i dati che gli vengono forniti, ma tra le sue funzioni vi è anche quella di controllare le unità interessate nell'eleborazione.

Tramite il bus di controllo, il microprocessore indicherà all'unità di memoria che vuole *leggere* il contenuto della posizione indirizzata; la parola dell'informazione passerà nel bus dei dati e da questo all'interno del registro delle istruzioni del microprocessore. Questa prima fase del trattamento dell'istruzione è definita *fase iniziale o inizio.*

Nella seconda fase di *decodificazione,* l'unità di controllo decodifica e interpreta il codice dell'operazione depositato nel registro delle istruzioni.

Infine, l'unità di controllo genererà i comandi adeguati nella fase definita di *esecuzione.*

Se l'esecuzione dell'istruzione necessita di più bit, l'unità di controllo verrà incaricata di fornirli e, in seguito, si passerà ad eseguire una nuova istruzione. Ogni volta che si accede alla memoria (fase iniziale) inizia un *ciclo macchina,* che contiene da 2 a 5 stati interni, ognuno dei quali corrisponde ad una microistruzione.

Esecuzione di una istruzione particolare

Per studiare il flusso dell'informazione e i segnali generati dall'unità di controllo del microprocessore, si descrive il processo di esecuzione di un'istruzione di caricamento. Per semplificare la descrizione si suppone che il microprocessore sia da 8 bit, con un bus degli indirizzi a 16 bit.

Istruzione di caricamento o lettura

Legge il contenuto della locazione di memoria indirizzata e lo carica nell'accumulatore. Questa istruzione si completa normalmente in quattro cicli di macchina.

- Primo ciclo di macchina:

ha il compito di riconoscere e decodificare il codice dell'operazione. Con il primo impulso di clock, il contenuto del contatore di programma è inviato al bus degli indirizzi.

Con il secondo impulso di clock si invia attraverso il bus di controllo un ordine di lettura e si incrementa di un'unità il contatore di programma.

Con il terzo impulso di clock, il contenuto della locazione di memoria è inviato al registro delle istruzioni tramite il bus dei dati.

Dopo il quarto impulso di clock, il contenuto del registro delle istruzioni verrà decodificato. La configurazione binaria del codice operativo indica che le due locazioni di memoria seguenti contengono l'operando (a 16 bit), che coincide con l'indirizzo in cui si trova il dato interessato all'esecuzione dell'istruzione.

- Secondo ciclo di macchina:

carica i primi otto bit dell'operando (indirizzo del dato) nella prima parte del registro di indirizzamento.

Si esegue nuovamente un'operazione di lettura in memoria e si deposita la parola binaria letta, nella prima parte del registro di indirizzamento.

- Terzo ciclo di macchina:

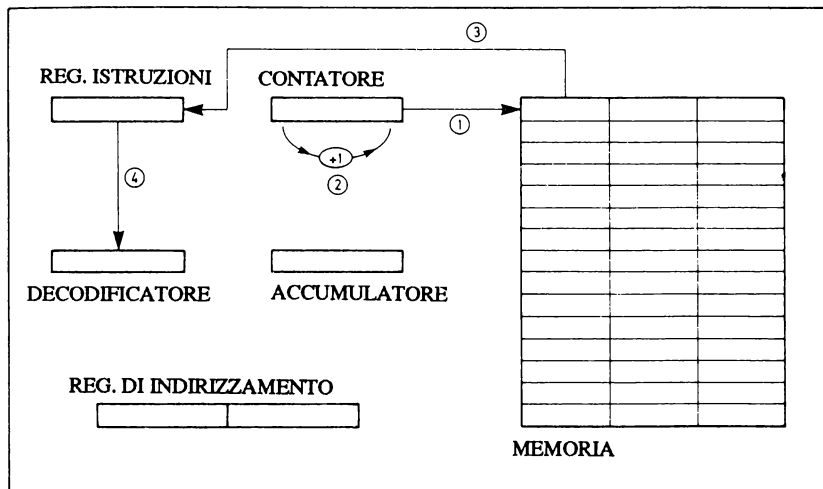
carica il secondo byte dell'operando (indirizzo del dato) nella seconda parte del registro di indirizzamento.

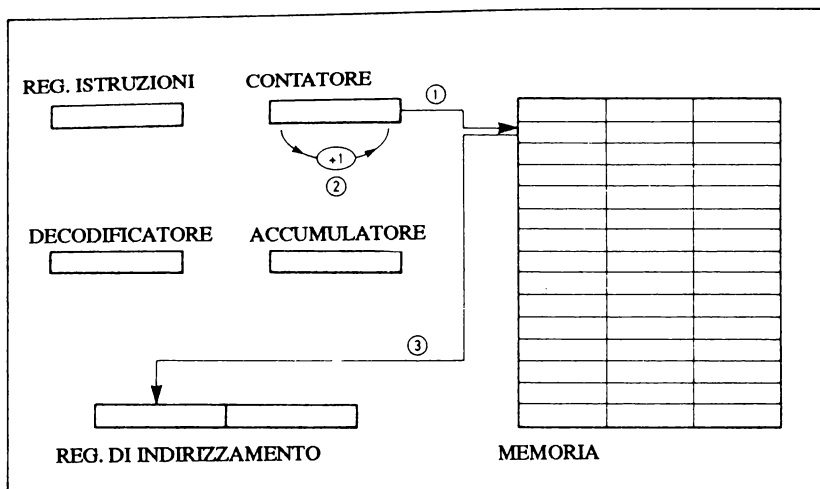
Si eseguono le operazioni analoghe a quelle del secondo ciclo macchina, eccetto che la parola binaria viene caricata nella seconda parte del registro di indirizzamento.

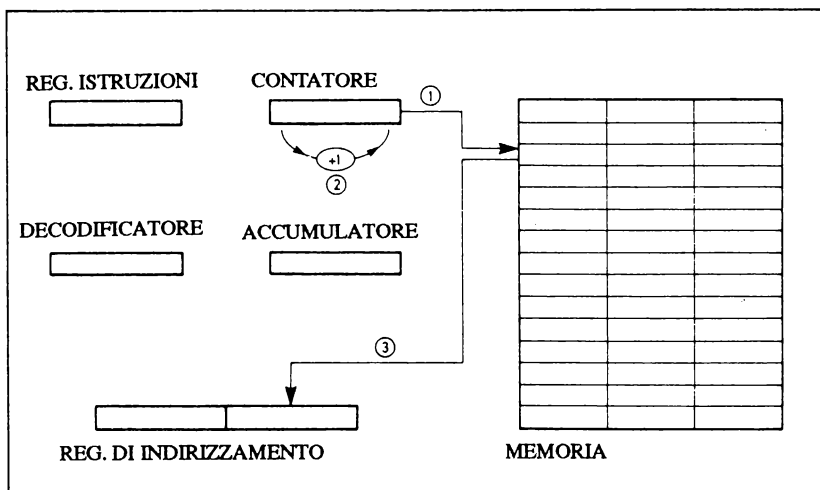
- Quarto ciclo di macchina:

in tale ciclo si effettua il caricamento del dato reale (il cui indirizzo era nell'operando), nell'accumulatore.

Con il primo impulso di clock si invia il contenuto del registro degli indirizzi.

Il secondo impulso coincide con la lettura del dato e la sua trasmissione al bus dei dati.

PRIMO CICLO ISTRUZIONI DI CARICAMENTO

IMPULSO DI CLOCK	AZIONE
①	INDIRIZZAMENTO
②	INCREMENTO CONTATORE
③	CARICAMENTO NEL REGISTRO DI ISTRUZIONE
④	DECODIFICA

SECONDO CICLO ISTRUZIONI DI CARICAMENTO

IMPULSO DI CLOCK	AZIONE
①	INDIRIZZAMENTO
②	INCREMENTO CONTATORE
③	CARICAMENTO NELLA 1ª PARTE DEL REG. INDIRIZZAMENTO

TERZO CICLO ISTRUZIONI DI CARICAMENTO

IMPULSO DI CLOCK	AZIONE
①	INDIRIZZAMENTO
②	INCREMENTO CONTATORE
③	CARICAMENTO NELLA 2ª PARTE DEL REG. INDIRIZZAMENTO

QUARTO CICLO ISTRUZIONI DI CARICAMENTO

IMPULSO DI CLOCK	AZIONE
1	INDIRIZZAMENTO
2	ACCESSO ALLA POSIZIONE DI MEMORIA
3	CARICAMENTO NELL'ACCUMULATORE

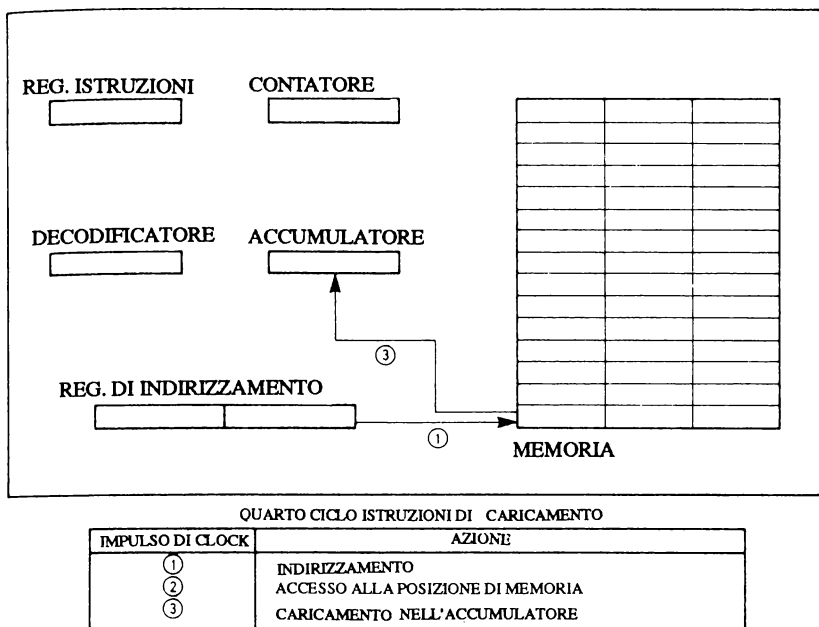



Fig. 2.-a) Primo ciclo dell'esecuzione di un'istruzione di caricamento. b) Secondo ciclo dell'istruzione. c) Operazioni eseguite durante il terzo ciclo. d) Quarto ciclo dell'istruzione.

Con il terzo impulso di clock, l'informazione contenuta nel bus dei dati viene trasferita nell'accumulatore, e nello stesso tempo si azzera il registro delle istruzioni, che rimane libero per ricevere l'istruzione successiva.

CAPITOLO 3

TECNOLOGIE COSTRUTTIVE

n questo capitolo, vengono descritte le diverse tecnologie utilizzate per la costruzione dei microprocessori integrati. Inoltre, sono evidenziate le varie tecniche che tendono a migliorare le caratteristiche elettriche di questi sottosistemi.

Introduzione

I microprocessori (µP) sono sottosistemi digitali relativamente complessi e, pertanto, adatti ad essere costruiti con la tecnologia di integrazione a larga scala (LSI).

Il primo µP integrato venne messo in commercio nel 1972. Nonostante il breve tempo trascorso, la ricerca nel campo dei semiconduttori è stata talmente intensa, da poter dire di essere giunti attualmente alla terza generazione. Senza dubbio, nei prossimi anni, a causa della grande concorrenza, verranno prodotte versioni migliori (maggior capacità, maggior rapidità, minor consumo, software più semplice, minor costo, ecc.).

Anche se si possono progettare versioni a componenti discreti più flessibili, ma più costose, con porte logiche standard, saranno descritte solo le tecnologie utilizzate attualmente per le versioni integrate. Saranno inoltre commentate le tendenze future prevedibili.

Nella tabella I sono riassunte le attuali tecnologie dei µP, evidenziando alcuni interessanti dati; nella tabella II sono invece riassunti i dati più interessanti relativi ai microprocessori.

Dall'esame comparativo delle due, si può ricavare una valida informazione sulla progettazione di questi componenti.

TABELLA I. ATTUALI TECNOLOGIE DEI μP

TECNOLOGIA	*FAMIGLIE*	*RITARDO PROPAGAZIONE PORTA*	*RITARDO × DISSIPAZIONE*	*TEMPO DI CICLO ISTRUZIONE*	*4 PORTE*			
					COMPO-NENTI	*SUPERFICIE (mm^2)*	*MASCHERE*	*DIFFUSIONI*
MOS	**CANALE P** 1972	< 1 μs (tip. 100 ns)	50	1,25 - 62,5 μs	2	10,6	4	1
	CANALE N PORTA AL SILICIO 1973	1 - 100 ns	1	0,3 - 13 μs	2	5,6	7	3
	CMOS 1974	< 100 ns (tip. 50)	4	0,5 - 5 μs	3	49,8	6	3
BIPOLARE	**TTL-SCHOTTKY BASSA DISSIPAZIONE (LSTTL)** 1974	10-25 ns	10-20	100-200 ns	3	19,9	7	4
	I^2L IMPIANTAZIONE IONICA 1975	10-50 ns	0,1-1	100 - 1000 ns	1	4,8	4	2
	ECL 1975	1 10 ns	50	55 ns	3	31	7	4

* 1 mil = 25,4 μ

TABELLA II. MICROPROCESSORI

FABBRICANTE MODELLO	DATI TECNOLOGICI	BITS IN PARALLELO	CAPACITA' INDIRIZZAMENTO	TEMPO DEL CICLO DI ISTRUZIONE (minimo)	ALIMENTAZIONE
Intel					
4040	PMOS	4	4 K	10,8 μs	15 ó – 10 y 5 V
8080	NMOS	8	64 K	1,5 μs	–5,5 y 12 V
8085	NMOS	8	64 K	1,3 μs	5 V
8036	HMOS	16	1 M	0,4 μs	5 V
Signetics					
2650	NMOS	8	32 K	2,4 μs	5 V; 525 mW
Motorola					
6800	NMOS	8	64 K	1 μs	5 V
68000	NMOS	16	16 M	0,5 μs	5 V
Rockwell					
6502	NMOS	8	64 K	2 μs	5 V
National					
PACE	PMOS	16	64 K	2,5 μs	–12,5 y 8 V
SC/MP	PMOS	8	64 K	5 μs	-7, +5 V
Zilog					
Z-80	NMOS	8	64 K	1 μs	5 V
Z 8000	NMOS	16	48 M	0,75	5 V

Riassunto delle famiglie utilizzate nei μP attuali

In questo paragrafo sono riassunte le varie famiglie utilizzate nei μP integrati.

PMOS

a) Porta statica

In Fig. 1 è visibile una porta NAND (logica negativa) con transistori MOS a canale P.

Tutte le porte MOS digitali sono costituite da transistori del tipo ad accumulazione. Il motivo è duplice: per permettere l'accoppiamento diretto tra stadi, e per minimizzare il consumo (se V_{GS}=0, I_D=0).

In tutte le famiglie MOS si utilizza come carico un transistore MOS che funziona come resistenza. In tal modo si riduce la superficie richiesta per l'integrazione. Il terminale di gate (G) del transistore di carico, perchè funzioni come tale, è solitamente connesso ad una tensione fissa (V_{GG}). Se V_{GG}=VP_{DD}, il transistore di carico è polarizzato nella sua regione piana di saturazione (il MOS è in saturazione se $|V_{DS}| > |V_{GS}-V_{TH}|$), il che non è adatto per la carica rapida delle capacità. Se V_{GG} raggiunge un potenziale fisso di grandezza superiore a V_{DD}, il transistore di carico può funzionare nella regione lineare, a minor impedenza. La connessione dipende dalla particolare applicazione.

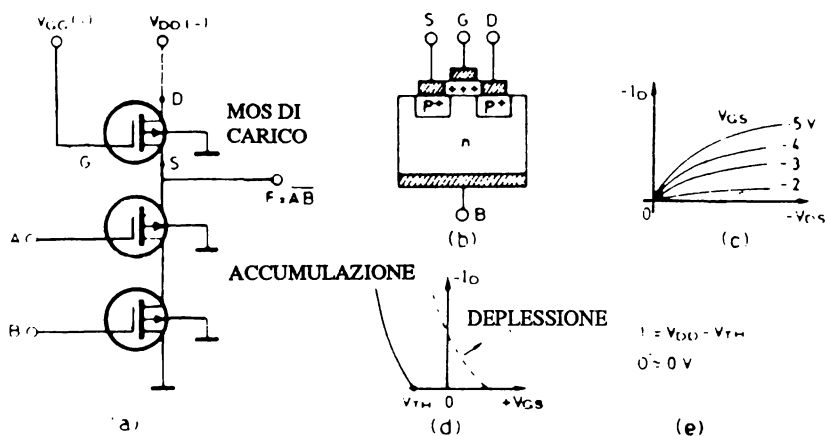



Fig. 1.-a) Porta NAND. b) Sezione trasversale. c) Caratteristiche del drain. d) Caratteristica di trasferimento. e) Livelli logici.

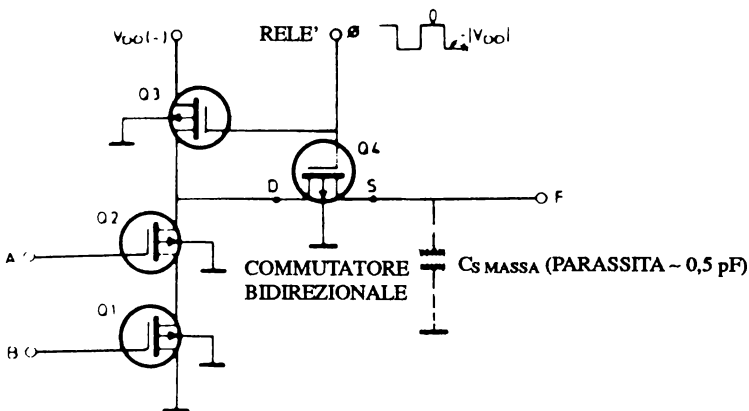



Fig. 2.-Porta MOS dinamica NAND ad una fase.

b) Porta dinamica

Le porte MOS dinamiche immagazzinano temporaneamente l'informazione in una capacità parassita. Il loro funzionamento è simile a quello dei circuiti di campionamento e mantenimento. Mediante un segnale di clock, l'immagazzinamento citato può essere reso permanente (operazione di refresh). Poichè le fughe sono molto piccole, le costanti di tempo di scarica delle capacità parassite sono, di solito, dell'ordine dei millisecondi e quindi il periodo del segnale di clock deve essere, tipicamente, inferiore a 1 msec (frequenza superiore a 1 KHz), per mantenere i dati permanentemente memorizzati.

In Fig. 2 si vede una porta MOS dinamica NAND.

Quando Φ=0, Q3 e Q4 sono interdetti, poichè non si forma canale, e rendono indipendente l'uscita (F) dal circuito di eccitazione, evitando inoltre il consumo della sorgente di alimentazione (V_{DD}).

Se Φ=1, Q3 e Q4 sono in conduzione e all'uscita si ottiene la funzione NAND.

Le porte dinamiche possono utilizzare più fasi (vari clock).

Il vantaggio dei circuiti dinamici rispetto a quelli statici è il minor consumo e costo.

Tra i vantaggi della famiglia PMOS si possono evidenziare:

- costruzione semplice (1 diffusione - 4 maschere);
- media densità di integrazione.

Gli svantaggi sono:

- grande lentezza;
- incompatibilità con altre famiglie logiche (necessitano interfacce).

Anche se la famiglia PMOS è stata la prima, sembra che il suo sviluppo futuro sarà scarso.

NMOS

Le porte fondamentali NMOS sono totalmente simili alle PMOS, varia solo la polarità dell'alimentazione.

I vantaggi degli NMOS rispetto ai PMOS sta semplicemente nella maggior velocità intrinseca di commutazione. Dato che i processi fisici sono dovuti al trascinamento dei portatori maggioritari e la mobilità degli elettroni è superiore a quella delle lacune, le risposte (tempi di commutazione) sono più rapide.

Anche se alcuni costruttori pensano che entro pochi anni i produttori di NMOS passeranno alla tecnologia I^2L, essendo questa meno costosa, attualmente lo sviluppo degli NMOS è molto elevato, grazie ai numerosi affinamenti che sono stati introdotti, e che ne migliorano sensibilmente le caratteristiche. I popolari microprocessori 8080 e MC6800 vengono costruiti con questa tecnologia.

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

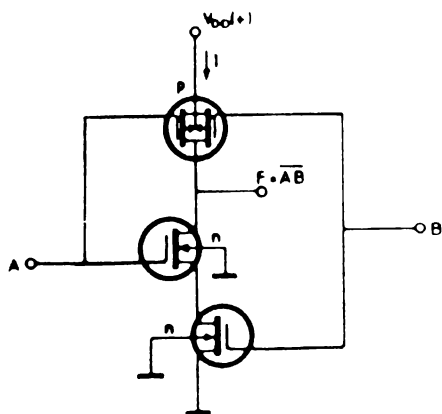



Fig. 3.-Porta NAND con CMOS.

La porta fondamentale NAND, a transistori complementari MOS è visibile in Fig. 3.

In Fig. 4 sono rappresentate le forme d'onda di uscita e della corrente in funzione dei segnali di ingresso.

Si osservi che circola corrente e, di conseguenza, si assorbe potenza dal-

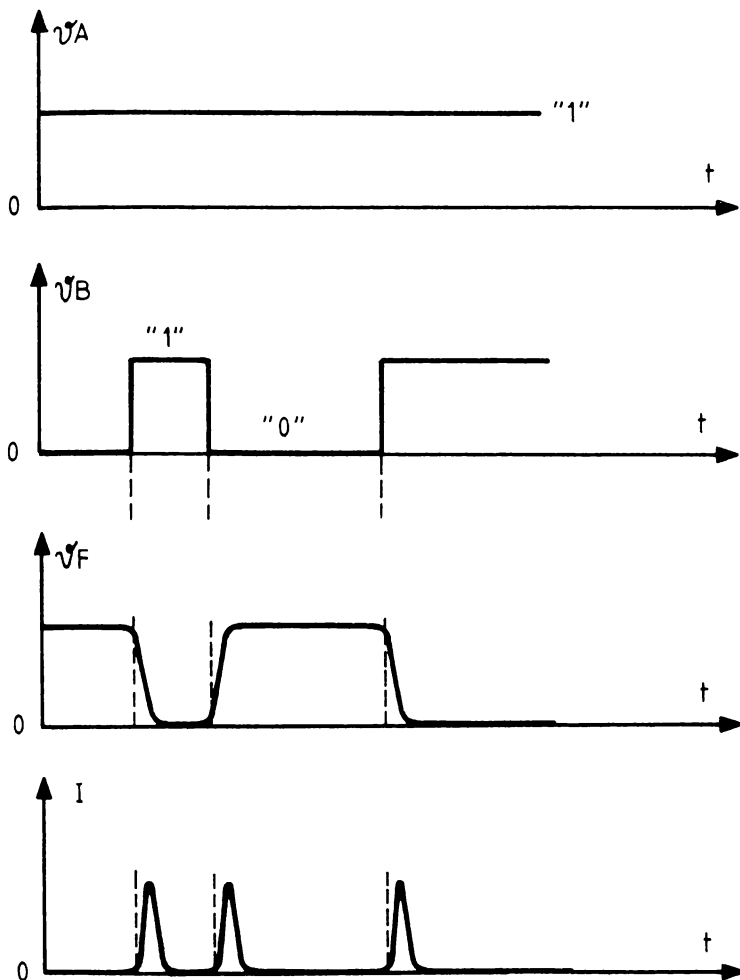



Fig. 4.-Diverse forme d'onda della porta NAND CMOS.

l'alimentazione, solo durante la transizione tra gli stati. In condizioni statiche il consumo, salvo le correnti di fuga, è nullo, in quanto i MOS tipo P, o gli N, sono interdetti.

Tra i vantaggi più rilevanti della famiglia CMOS si evidenziano:

- piccola dissipazione;
- ampio campo di alimentazione (tipicamente da 3 a 15 V non stabilizzati);
- ampi margini di rumore.

Gli svantaggi sono:

- senza tecniche raffinate occupano molto spazio sul chip;
- il processo di costruzione è relativamente complesso.

TTL-SCHOTTKY

La TTL/S è una famiglia non saturata, e pertanto con elevata velocità di commutazione. Per non saturare i transistori viene integrato un diodo Schottky tra base e collettore (Fig. 5).

Poichè l'allumino è un'impurezza accettrice (gruppo III), quando viene depositato su silicio di tipo N tende a formare una barriera metallo-semiconduttore, invece di un contatto ohmico. Pertanto il processo tecnologico di costruzione è relativamente semplice. La caduta di tensione con polarizzazione diretta è di circa 0,4 V e, poichè non si accumulano cariche nel diodo, questo risulta essere molto veloce.

Il circuito TTL/S ad alta velocità contiene, in tutti i transistori che si sa-

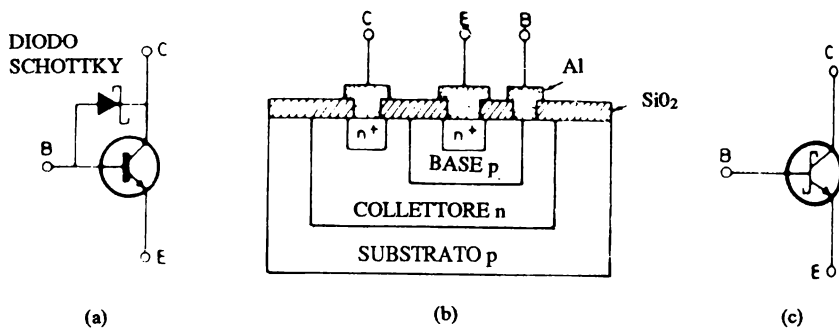



Fig. 5.-a) Transistore NPN con diodo Schottky di blocco. b) Sezione trasversale. c) Simbolo.

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

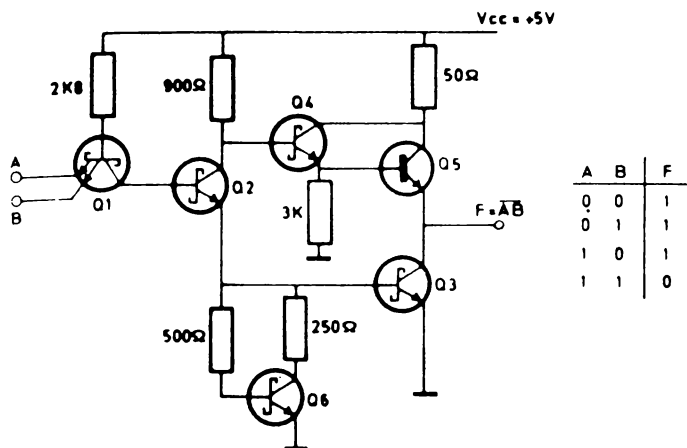



Fig. 6.-TTL-Schottky (NAND).

turano nel TTL standard, un diodo Schottky (Fig. 6).

Siccome Q5 non si può saturare, il diodo non è necessario.

Il principale inconveniente del TTL/S è l'elevata dissipazione; per questo motivo, e per competere con i CMOS, è stata sviluppata una versione leggermente diversa a bassa dissipazione (con perdita di velocità), detta Low Power Schottky TTL (LSTTL) (a basso consumo), molto utilizzata nei µP bipolari. Analogamente alle altre famiglie, è disponibile un'uscita addizionale ad alta impedenza, che caratterizza le versioni a tre stati (*0, 1* e ad alta impedenza).

Tra i vantaggi vi sono:

- elevata velocità;
- numerose periferiche in tecnologia TTL.

Inconvenienti:

- processo costruttivo complesso;
- necessità di sorgenti di alimentazione stabilizzate (5±0,5 V);
- dissipazione elevata.

I²L (Integrated Injection Logic)

La famiglia I²L (detta anche MTL-Merged Transistor Logic) (Fig. 7) derivata dalla DCTL, attualmente in disuso, presenta i vantaggi di una elevata

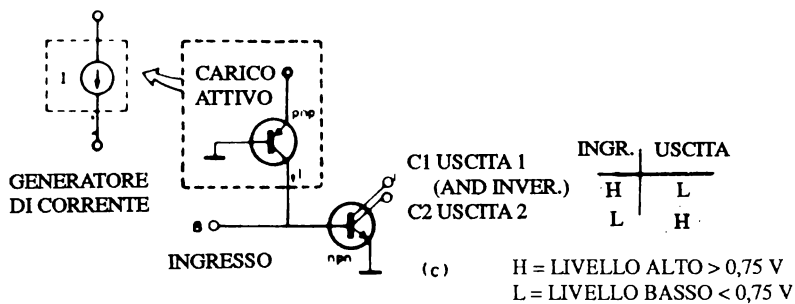
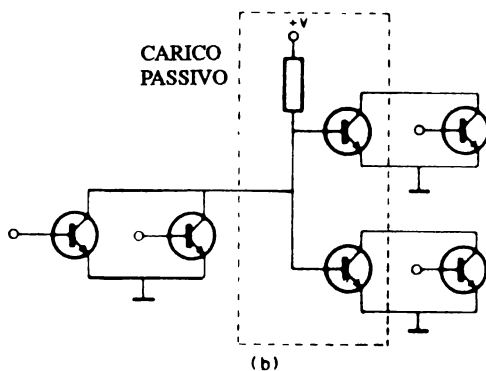
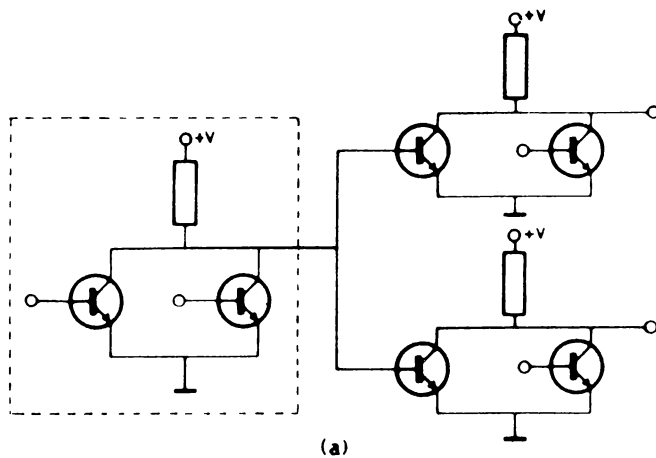
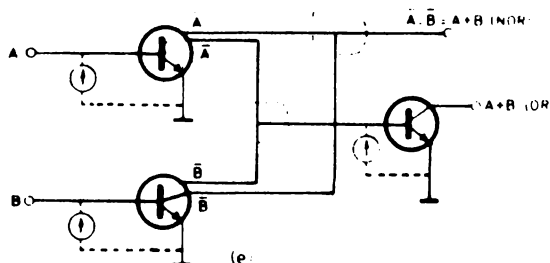
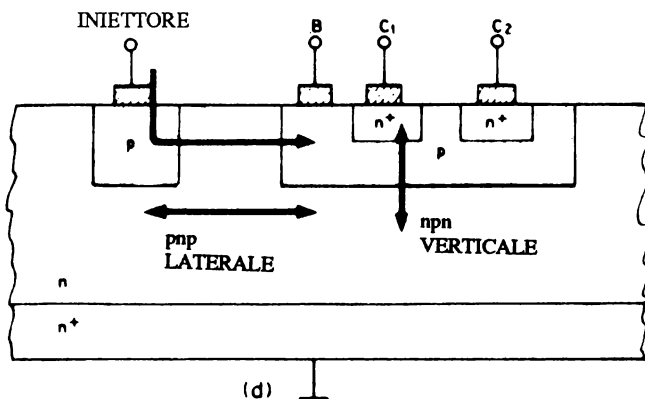



Fig. 7.-Logica ad iniezione integrata. a) Struttura DCTL. b) Transizione verso I^2L. c) Porta fondamentale I^2L (invertitore) in logica positiva. d) Sezione trasversale tipica. e) Esempio per generare le funzioni OR e NOR.

densità di integrazione, semplice processo di costruzione, e basso consumo.

Le sorgenti di corrente, i transistori laterali PNP, funzionano come carico. I transistori verticali NPN si comportano come invertitori.

Le porte I^2L vennero presentate nel 1972. Sia in DCTL che in I^2L, i transistori funzionano come invertitori, e le funzioni logiche (AND cablato) si ottengono collegando i collettori.

Non necessitando di resistenze integrate, la I^2L richiede poco spazio, per cui le capacità parassite sono molto piccole, e, essendo il salto logico ridotto (V_{BEON}-V_{CEsat}≈0,6 V), tali capacità si caricano e scaricano rapidamente, il che spiega il basso valore del fattore di merito ritardoxdissipazione.

In Fig. 7d) si può osservare che il funzionamento tipico del transistore integrato NPN in I^2L è opposto a quello dei transistori abituali, cioè l'iniezio-

ne di minoritari nella base si dirige verso la superficie. Fondamentalmente:

- il transistore NPN è saturato quando la sua base è fluttuante, cioè in circuito aperto ($V_{BE} \approx 0{,}6$ V);
- il transistore NPN è interdetto quando la sua $V_{BE}=0$ V. (I_B andrà a massa e $I_C=0$).

La tavola della verità è visibile in Fig. 7c). La cellula isolata si comporta come un invertitore, dato che le uscite sono complemento dell'ingresso.

In Fig. 7e) è visibile un esempio di come si possano generare le funzioni OR e NOR, in logica positiva.

La famiglia I^2L è compatibile TTL. La tensione di alimentazione deve essere superiore a 0,85 V. Tipicamente il campo va da 1 a 15 V.

Si pensa che presto si raggiungeranno ritardi di propagazione di porta dell'ordine di 1 nsec, il che costituirà una forte concorrenzialità con la tecnologia attualmente più veloce: la ECL, che è quella dal consumo più elevato. Tra le migliorie che si stanno sperimentando per rendere più rapida la I^2L ci sono:

- impiantazione ionica;
- diodo Schottky;
- isolamento passivo.

ECL (emitter-coupled-logic)

E' la più rapida di tutte le famiglie attualmente esistenti, in quanto i suoi transistori non entrano mai in saturazione.

Anche se ne esistono altre leggermente diverse, la porta fondamentale ECL è quella visibile in Fig. 8.

Il funzionamento del circuito è molto semplice. Siccome:

$$V_R = \frac{0+1}{2}$$

se l'ingresso è *0*, Q1 sarà interdetto e Q2 funzionerà nella regione attiva; se l'ingesso è *1*, la situazione è opposta. Le tensioni sui collettori di Q1 e Q2 giungono all'uscita tramite gli inseguitori di emettitore.

Tra i vantaggi si possono citare:

- elevata velocità;
- basso rumore.

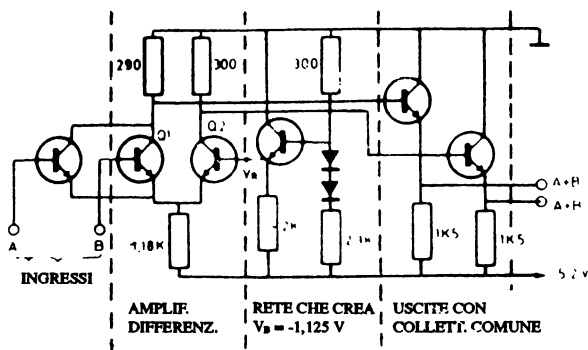



Fig. 8.-Porta fondamentale ECL. Le uscite sono funzioni OR e NOR degli ingressi.

Tra gli inconvenienti:

- elevata dissipazione;
- incompatibilità con le altre famiglie;
- processo di fabbricazione complesso;
- ridotta densità di integrazione.

Anche se sono state sinteticamente descritte le famiglie principali attualmente utilizzate nei µP, occorre non dimenticare che si stanno studiando nuove strutture di grande potenzialità per l'applicazione nei µP. A titolo di esempio, si citano le seguenti:

- C^3L (Complementary Constant-Current Logic). Logica a corrente costante complementare;
- T^3L (Incorpora uno stadio separatore aggiuntivo);
- Schottky I^2L;
- 3D EFL (Triple-Diffused Emitter-Follower Logic). Logica ad inseguitore di emettitore a tripla diffusione.

Miglioramenti tecnologici

Gli sforzi tecnologici sono incessantemente diretti a trarre il massimo vantaggio dalla tecnica LSI. I miglioramenti che vengono sperimentati sono

molto vari, ma hanno tutti i seguenti obiettivi:

- aumentare la densità di integrazione (strutture semplici e compatte);
- aumentare la velocità di commutazione (diminuzione delle capacità intrinseche e parassite, miglior allineamento);
- sorgenti di alimentazione semplici (unica alimentazione, bassa tensione non stabilizzata, tensione variabile, ecc.);
- massima immunità (al rumore, alle radiazioni, alle variazioni di temperatura...);
- minimo numero di terminali (interfacce semplici);
- maggior affidabilità;
- compatibilità con le altre famiglie logiche;
- maggior numero di circuiti senza difetti (maggior rendimento);
- prezzo ridotto.

La contropartita sta nella maggior complessità di costruzione e, in definitiva, nel costo.

CAPITOLO 4

MICROPROCESSORI A 8 BIT PIU' COMUNI (I)

n questo capitolo saranno evidenziate le più importanti caratteristiche di alcuni microprocessori a 8 bit della INTEL, molto utilizzati.

Il microprocessore 8080

Il microprocessore 8080 fu commercializzato dall'INTEL nel 1974; in realtà si tratta di una versione migliorata dell'8008 che è stato la prima CPU integrata a 8 bit. L'8080 è stato uno dei microprocessori di maggior attualità e, pertanto, dei più utilizzati.

Caratteristiche dell'8080

Le sue caratteristiche più salienti sono:

- microprocessore a 8 bit;
- costruito con tecnologia MOS a canale N;
- capacità di indirizzamento di 64 Kbytes;
- frequenza tipica di clock: 500 KHz;
- 74 istruzioni di programmazione;
- possibili modi di indirizzamento: diretto esteso, immediato, implicito, diretto e indiretto tramite registro;
- alimentazione che necessita di tre diverse tensioni rispetto a

Foto 1.-Chip 8084 prodotto da INTEL e NEC, con caratteristiche simili a quelle esposte nel testo.

massa: +5 V, -5 V, e +12 V;

- sistema di interrupt: possiede un terminale su cui si configurano tutte le richieste di interrupt, con 8 livelli di priorità.

Struttura interna dell'8080

Come indicato in Fig. 1, il µP 8080 dispone di dieci registri interni per la sua programmazione.

- Un registro accumulatore a 8 bit, detto A, che è un registro di uso generale, attraverso cui entrano ed escono i dati esterni.
- Sei registri generali a 8 bit, che si possono considerare come sei registri da otto bit ciascuno, o come tre registri da sedici bit ciascuno. Il fatto di considerarli a otto o sedici bit viene determinato da ogni istruzione; esistono istruzioni che considerano i sei registri indipendentemente, e altre che li raggruppano a due a due per formare registri a sedici bit. Tali

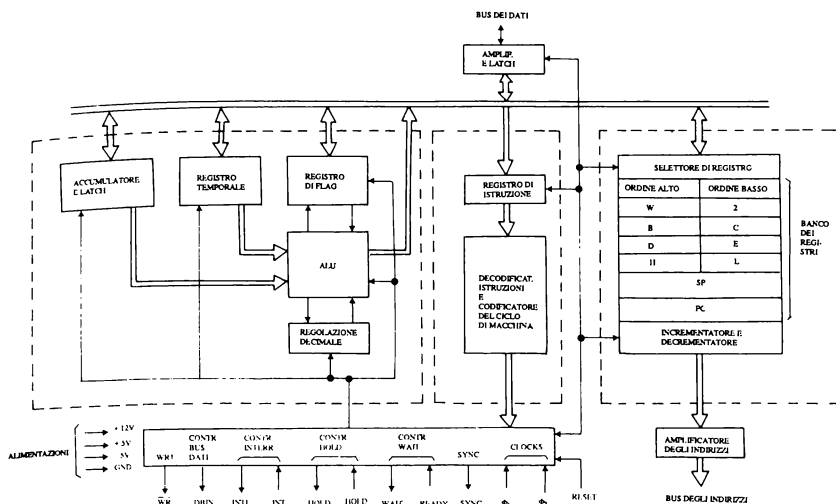



Fig. 1.-Diagramma a blocchi dell'8080.

registri si chiamano B, C, D, E, H, L. Le istruzioni che necessitano registri a sedici bit, li raggruppano a due a due nel modo seguente: B con C, D con E, e H con L.

- Un puntatore di stack a sedici bit per l'annotazione automatica dell'indirizzo di rientro dalle subroutines, detto SP.
- Un contatore di programma a sedici bit per indicare l'indirizzo dell'istruzione da eseguire, detto PC.
- Un registro di stato a otto bit che, come in tutti i microprocessori, rappresenta lo *stato,* e nel quale ognuno degli otto bit ha un proprio significato, il cui contenuto è relativo alle condizioni di funzionamento del µP in un dato istante. Il significato di ognuno di questi otto bit, rappresentati in Fig. 2, è il seguente:
  - Bit 0: carry, questo bit indica che il risultato dell'esecuzione dell'istruzione ha un riporto; quando esiste tale riporto il bit è zero.
  - Bit 1: non utilizzato, è sempre uno.
  - Bit 2: parità, questo bit indica se il dato elaborato è pari o dispari, quando il bit vale 1, il dato è dispari, se zero, pari.
  - Bit 3: non utilizzato, è sempre zero.
  - Bit 4: carry intermedio, questo bit indica il riporto tra mezzi bytes, tra il bit 3 e il 4 di un dato. Se un byte viene suddiviso

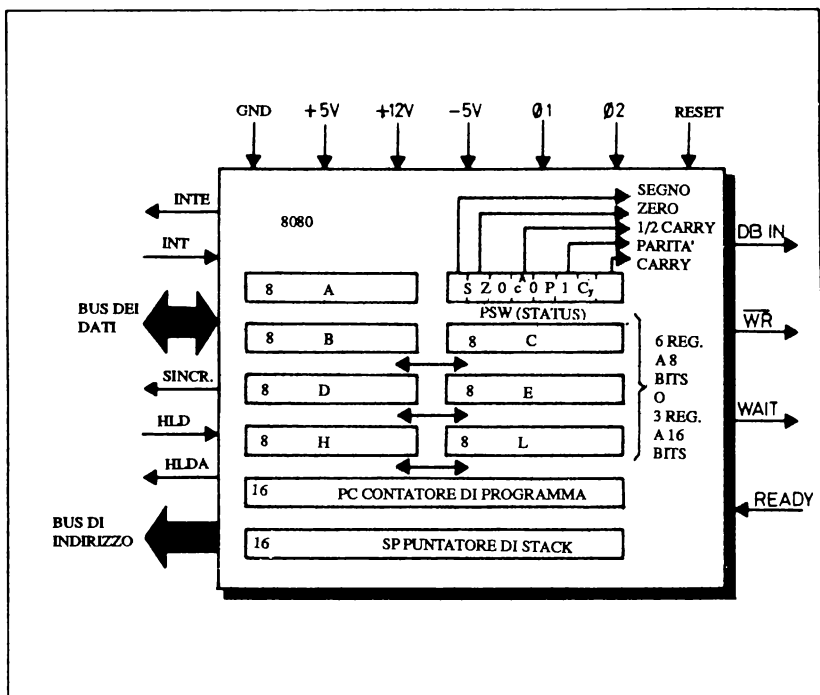



Fig. 2.-Registri interni del microprocessore 8080.

in due, si hanno due parti da 4 bit, ognuna di queste parti è in grado di rappresentare un esadecimale da 0 a F. Esistono casi in cui è necessario conoscere quando si passa da F a 0, incrementando un dato a 8 bit. Questo fatto è indicato dal bit di carry intermedio, che in tal caso assume valore 1.

- Bit 5: non utilizzato, è sempre zero.
- Bit 6: zero, questo bit vale zero se il risultato ottenuto in una operazione vale zero, altrimenti il bit vale 1.
- Bit 7: segno, indica il segno del risultato del dato elaborato; se il dato è negativo, il bit vale 1. (In esadecimale i numeri da 80 a FF sono negativi, cioè lo sono tutti quei dati in cui il bit 7 vale 1), mentre se il risultato è positivo (in esadecimale da 00 a 7F), il bit di segno è 0.

Funzione dei terminali

In Fig. 3, dalla quale si può rilevare il numero del piedino che corrisponde a ciascuno dei 40 di cui dispone il microprocessore o chip, è indicata la distribuzione dei segnali sui terminali.

La funzione di ogni terminale è la seguente:

- A0-A15, da questi terminali esce verso l'esterno il bus degli indirizzi composto da 16 bit, per coprire il campo di 64 Kbytes di spazio indirizzabile.
- D0-D7, questi terminali costituiscono il bus dei dati attraverso cui circola il flusso di dati, da e verso l'esterno, quindi bidirezionale.

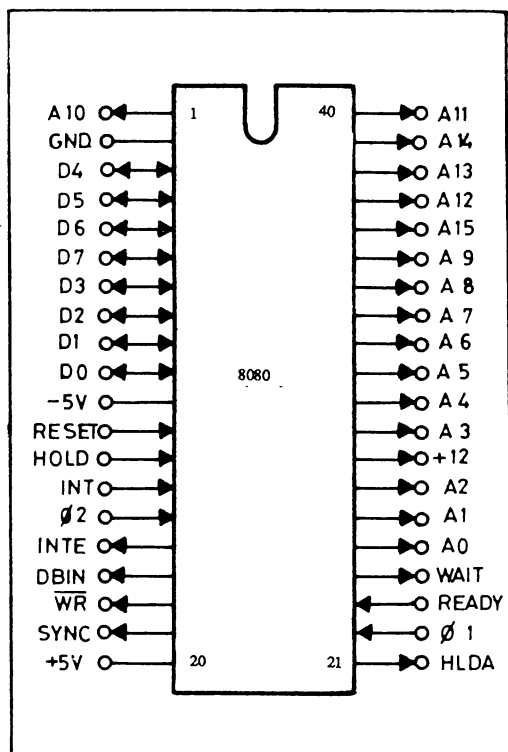



Fig. 3.-Distribuzione delle funzioni sui terminali del μP 8080.

- RESET, segnale di azzeramento di tutti i registri interni del µP. Ciò avviene quando questo terminale è ad uno per almeno tre cicli di clock.
- HOLD, quando questo segnale di ingresso diventa 1, i bus degli indirizzi e dei dati passano in alta impedenza, vengono cioè *scollegati,* in modo che un altro µP o qualche periferica li usi per il tempo necessario.
- HLDA, tramite questo terminale il µP comunica all'esterno che è stato inserito un HOLD e, pertanto, i bus non sono disponibili, essendo ad alta impedenza.
- INT, segnale di richiesta di interrupt; tramite il corrispondente terminale, le periferiche possono sollecitare un interrupt dall'esterno; ciò avviene quando il terminale si porta a uno.
- INTE, segnale di accettazione dell'interrupt; tramite il terminale corrispondente a questo segnale il µP indica alla periferia richiedente l'interrupt, che lo stesso è stato accettato, portandosi a uno logico.
- DBIN, attraverso questo terminale il µP indica ai circuiti esterni che il bus dei dati è in ingresso, cioè in condizione di lettura dati, portandolo a uno.
- WR, tramite tale terminale il µP indica ai circuiti esterni che il bus dei dati è in scrittura, fatto che avviene quando il terminale diventa zero. I segnali DBIN e WR sono quelli che indicano alle periferiche e alla memoria se il µP è in lettura o scrittura.
- SYNC, terminale di sincronizzazione; il µP lo setta a uno quando inizia l'esecuzione di una nuova istruzione.
- ALIMENTAZIONE, il µP 8080 necessita di tre tensioni di alimentazione +12 V, +5V, e -5 V, rispetto al punto di riferimento, cioè alla massa.
- SEGNALI DI CLOCK, questo µP necessita di due segnali di clock Φ_1 e Φ_2, sfasati di 180 °; questi due terminali di clock non accettano livelli di tensione TTL, per cui occorre ricorrere a un circuito integrato come l'8224, progettato dal costruttore a tale scopo, per generare questi clock a partire da un oscillatore al quarzo.
- READY, segnale di pronto; settando a uno logico questo terminale, le periferiche o memorie lente, indicano al µP che sono pronte per mettersi in contatto con la CPU.

- WAIT, segnale di attesa; portando a uno logico questo terminale, il µP comunica all'esterno lo stato di attesa, che si verifica, ad esempio, quando si mette in contatto con una periferica lenta che ha fornito il segnale READY.

Sistema di collegamento dell'8080

Questo microprocessore non può essere direttamente collegato alla memoria e alle periferiche, ma necessita di due circuiti integrati esterni, progettati per interfacciarlo in alcuni dei suoi segnali e delle sue funzioni.

Questi circuiti sono: l'8224, utilizzato come generatore dei segnali di clock, reset e di altri sincronismi, e il circuito integrato 8228 utilizzato come amplificatore del bus dei dati, e come controllore di alcuni segnali, a cui viene fornita la forma adatta, per poter accedere alla memoria e alle periferiche.

Tutto il complesso, costituito dal µP e dai circuiti ausiliari è visibile in Fig. 6.

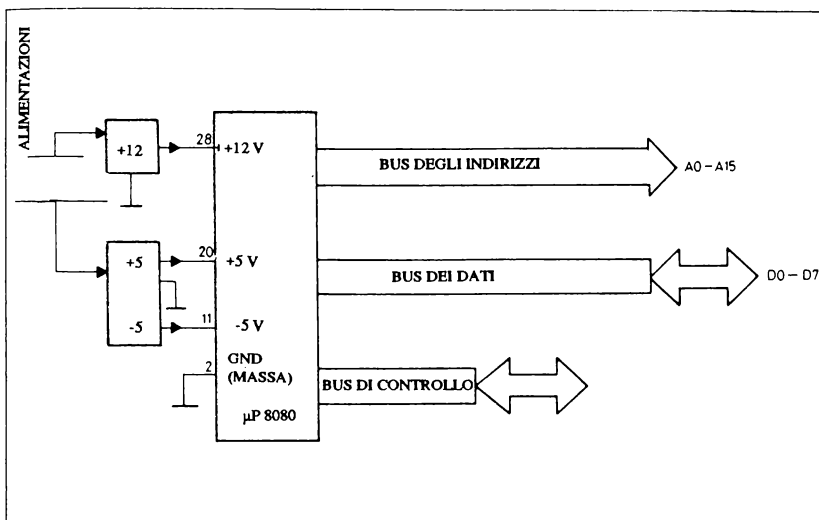



Fig. 4.-Dettaglio dei bus e delle alimentazioni del µP 8080.

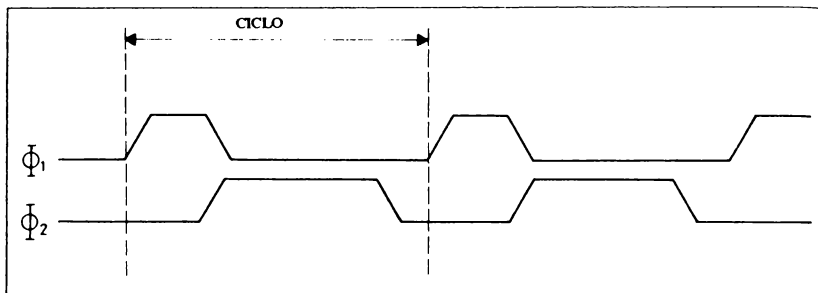



Fig. 5.-Dettaglio delle fasi del clock: Φ1 e Φ2.

8224, generatore di clock

Questo circuito integrato ha il compito di generare le due fasi di clock Φ1 e Φ2, a partire da un oscillatore interno, controllato esternamente da un cristallo di quarzo e da un circuito risonante LC, con il quale seleziona la frequenza fondamentale del quarzo, o un'armonica di tale fondamentale.

I valori assegnati a LC servono per aumentare il guadagno dell'armonica relativa alla frequenza adatta ai clock del microprocessore (Φ1 e Φ2); in tal modo il valore della frequenza del quarzo non è determinante.

Questo circuito integrato, inoltre, fornisce l'ampiezza minima del segna-

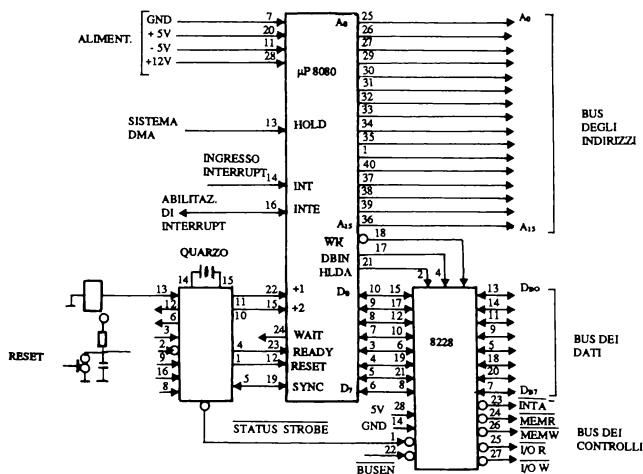



Fig. 6.-Schema dei collegamenti dell'8080 e dei suoi circuiti.

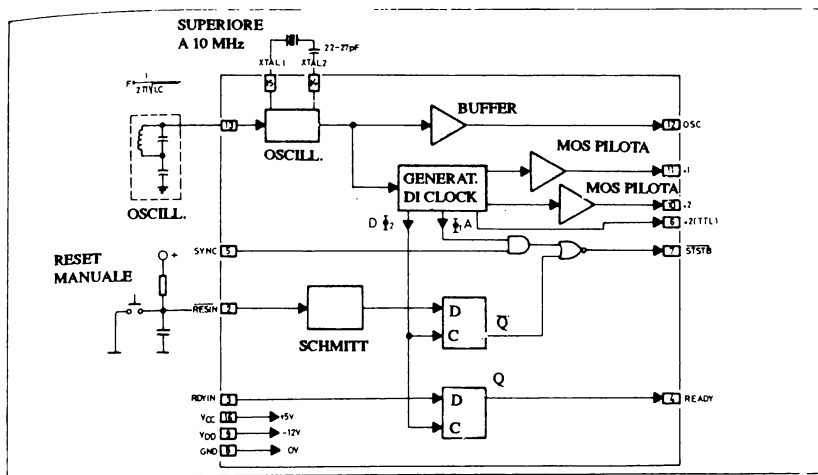



Fig. 7.-Schema interno a blocchi dell'8224.

le di reset del µP, a partire dal clock e dal reset manuale esterno.

Il circuito 8224 ha due altri compiti complementari, cioè fornire al µP il segnale di READY, in combinazione con il clock (Φ_2) ed il segnale di READY esterno.

Il secondo compito dell'8224 è di fornire il segnale STSTB (Status Strobe), che serve a dare passaggio allo *status* di funzionamento attraverso il bus dei dati verso il controller 8228, di cui si tratterà più avanti; il segnale STSTB è prodotto in combinazione con il segnale SYNC del microprocessore.

Si vedano le Figg. 6 e 7 per meglio comprendere quanto esposto.

8228, controller e amplificatore di bus

Questo circuito integrato ausiliare per il µP 8080, gli fornisce un controllo completo sia del bus dei dati, sia dei segnali che consentono l'accesso alle memorie ed agli ingressi ed uscite speciali.

L'8228 è suddiviso, a livello di funzionamento, in due parti perfettamente differenziate, la prima controlla il bus dei dati, in entrambi i sensi, tramite i segnali HLDA e DBIN, provenienti dal µP, per determinare lo stato ad alta impedenza e di flusso dei dati, sia in scrittura (W), sia in lettura (R). Nello stesso istante in cui avviene l'attivazione di tali controlli, i bit subiscono un'amplificazione di corrente che permette un maggior FAN OUT del bus dei dati verso l'esterno del microprocessore.

La seconda parte di questo circuito integrato riceve il contenuto del bus dei dati elaborato dal µP, nel periodo di tempo in cui esiste il segnale SYNC nel primo ciclo di ogni istruzione; tale contenuto è lo stato di certi bit interni che indicano lo stato funzionale del µP.

Il significato di tali bit è interpretato, mediante una logica interna, dall'8228, che attiva in base ad essi i propri bit di uscita per controllo della memoria e delle periferiche.

I bit forniti dal µP hanno il seguente significato: D0=INTA è il segnale di riconoscimento dell'interrupt, D1=WO indica il modo di lettura o scrittura, D2=STK indica che il contenuto dello stack è nel bus degli indirizzi, D3=HLTA indica che il µP si è arrestato, D4=OUT indica che si sta eseguendo un'istruzione di uscita, D5=M1 indica che il µP è nel ciclo di ricerca, D6=INP indica che si sta eseguendo un'istruzione di ingresso, e D7=MEMR indica un'operazione di lettura della memoria.

Uscite dell'8228

L'8228 esegue due compiti, il primo è di amplificare in entrambi i sensi il bus dei dati mediante i segnali di controllo WR, DBIN e HLDA; il secondo di generare i segnali di accesso alla memoria e alle periferiche prendendo l'in-

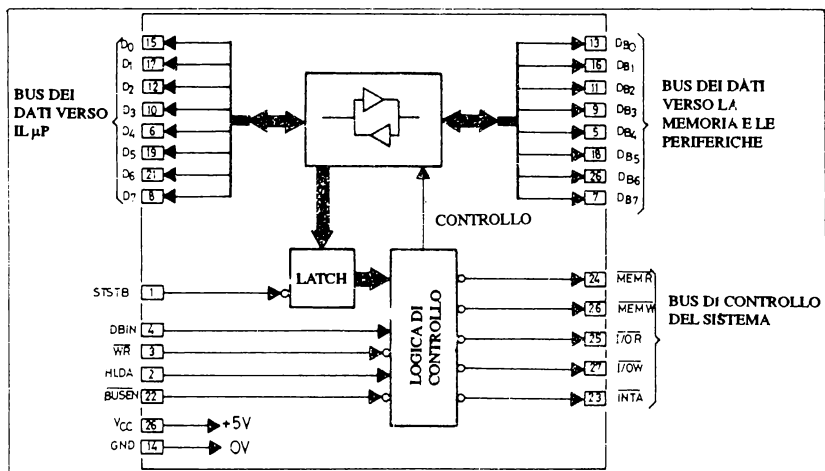



Fig. 8.-Schema interno a blocchi dell'8228.

formazione dal bus dei dati in un secondo ciclo tramite il segnale $\overline{\text{STSTB}}$, proveniente dal µP.

L'informazione fornita dal microprocessore attraverso il bus dei dati per questo secondo compito, attiva le uscite dell'8228, come è indicato nella tabella I. Queste uscite sono quelle che controllano e sincronizzano tutta la periferia della CPU.

- Il segnale $\overline{\text{MEMR}}$ indica alla memoria l'operazione di lettura (R=READ), è un impulso basso o negativo, mediante il quale la memoria fornisce il dato contenuto nella cella indicata dall'indirizzo del corrispondente bus.
- Il segnale $\overline{\text{MEMW}}$ indica alla memoria l'operazione di scrittura (W=WRITE), ed è anch'esso un impulso basso o negativo, mediante il quale viene scritto in memoria inserisce il dato presente sul bus dei dati, nella cella corrispondente all'indirizzo indicato sul bus degli indirizzi.
- Il segnale $\overline{\text{I/OR}}$ indica alle periferiche l'operazione di lettura; è un impulso negativo mediante il quale si trasferisce l'informazione da una periferica, ad esempio una porta (PORT), al microprocessore. Una porta è, in realtà, un insieme di LATCHES o bistabili progettati per essere collegati ad un µP.

BUS DEI DATI	NOME DEI BITS	CICLO DI RICERCA ①	MEM R. ②	MEM W. ③	STACK R ④	STACK W ⑤	R INGRESSO ⑥	W USCITA ⑦	INFA ⑧	HALT ⑨	HALT ⑩
D_0	INTA	0	0	0	0	0	0	0	1	0	1
D_1	$\overline{\text{WO}}$	1	1	0	1	0	1	0	1	1	1
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP	0	0	0	0	0	0	0	0	0	0
D_7	MEMR	1	1	0	1	0	0	0	0	1	0

Tabella I.-Interpretazione dei bits del bus dei dati attraverso l'8228 per produrre segnali di controllo della periferica.

- Il segnale $\overline{\text{I/OW}}$ indica alle periferiche l'operazione di scrittura, ed è un impulso negativo utilizzato per trasferire l'informazione dal µP alla periferica indirizzata.
- Il segnale INTA (Interrupt Acknowledge) è un'uscita che indica alla periferica, richiedente l'interrupt, che questo è stato accettato dal µP.
- Il segnale $\overline{\text{BUSEN}}$ (Bus Enable Input) è un ingresso tramite il quale si portano ad alta impedenza le uscite suddette, per bloccare i controlli.

Circuito applicativo dell'8080

Lo schema di un circuito applicativo è rappresentato nella Fig. 9, in cui sono indicati dettagliatamente i collegamenti dell'8080 ai suoi circuiti ausiliari, alla memoria, e alle periferiche.

Si osservi come l'8224 controlla il µP a partire dal clock generato da un quarzo, e dal reset ottenuto premendo il pulsante esterno. Si osservino anche i collegamenti dell'8228. Il bus dei dati si collega alla memoria, alle periferiche o all'esterno, mediante gli amplificatori interni; le uscite di lettura e scrittura, generate dall'8228, devono essere collegate alla ROM o EPROM, in cui si trova il firmware applicativo, alla RAM, alle porte e all'esterno.

Mentre i segnali di lettura e scrittura si collegano agli ultimi tre blocchi, quello di scrittura non viene inviato alla ROM, poichè quest'ultima può solo essere letta. I tre blocchi sono indirizzati da un decodificatore, i cui ingressi sono A13, A14 e A15, del bus degli indirizzi, in modo che l'indirizzo 0XXX attiva la ROM, 1XXX attiva la RAM, e 2XXX attiva la porta. I restanti indirizzi sono liberi.

Quando su un'uscita del decodificatore è presente, ad esempio, 2XXX significa che su tale uscita sono abilitati gli indirizzi compresi tra 2000 e 2FFF esadecimale. L'ingresso G del decodificatore è un ingresso di permesso, fornito dal microprocessore con il segnale HLDA, per ottenere il sincronismo tra periferiche e µP.

Tabella delle istruzioni dell'8080

Il set di comandi per lo sviluppo del software per il microprocessore 8080 è costituito da 74 istruzioni. Queste possono, a loro volta, essere formate da uno, due e tre bytes. Le operazioni che sono in grado di effettuare, sono quel-

Fig. 9.-Schema completo di un circuito applicativo con il μP 8080.

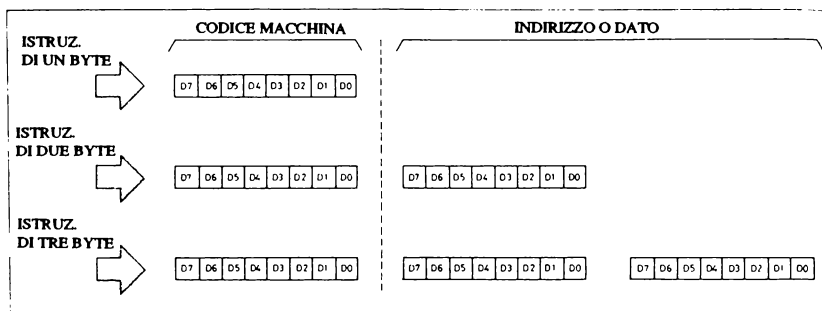



Fig. 10.-Formato delle istruzioni dell'8080. Il primo byte è occupato dal codice macchina o codice operativo, il secondo e terzo da un dato esadecimale o da un indirizzo.

le tipiche di quasi tutti i microprocessori.

Le loro denominazioni mnemoniche sono sigle delle funzioni in lingua inglese, di cui viene fornito un glossario.

Nome delle istruzioni

L'insieme delle 74 istruzioni dell'8080, descritte nella tabella II, sono suddivise in cinque parti, raggruppate per funzioni simili.

- Nel gruppo di trasferimento vi sono:

- le istruzioni di caricamento LOAD, che caricano nell'accumulatore i registri o il puntatore dello STACK;
- le istruzioni di movimento MOVE, che scambiano tra loro il contenuto dei registri;
- le istruzioni di memorizzazione STORE, che trasferiscono il contenuto dell'accumulatore e dei registri nella memoria.

- Nel gruppo delle istruzioni aritmetiche si hanno:

- le istruzioni di controllo dell'interrupt, che lo permettono o no;
- le istruzioni di somma ADD, che eseguono la somma, con o senza riporto, tra i diversi registri e con la memoria; l'istruzione DAA opera in BCD;
- le istruzioni di decremento e incremento, che effettuano trasferimenti incrementando o decrementando il contenuto di una unità;

TABELLA II

Mnemonico		Descrizione	Operazione	Codice macchina 7	6	5	4	3	2	1	0	Num. bytes M	Num. cicli T	Num. MC	Stato S	Z	AC	P	CY
Istruzioni di trasferimento																			
LDA		Load Accumulator Direct	(A) ← ((byte 3) (byte 2))	0	0	1	1	1	0	1	0	3	4	13					
LDAX	B	Load Accumulator Indirect	(A) ← ((B))	0	0	0	0	1	0	1	0	1	2	7					
LDAX	D	Load Accumulator Indirect	(A) ← ((D))	0	0	0	1	1	0	1	0	1	2	7					
LHLD		Load H and L Direct	(L) ← ((byte 3) (byte 2))	0	0	1	0	1	0	1	0	3	5	16					
			(H) ← ((byte 3) (byte 2)+1))																
LXI	B	Load Inmediate, Registers B and C	(B) ← (byte 3)	0	0	0	0	0	0	0	1	3	3	10					
			(C) ← (byte 2)																
LXI	D	Load Inmediate, Registers D and E	(D) ← (byte 3)	0	0	0	1	0	0	0	1	3	3	10					
			(E) ← (byte 2)																
LXI	H	Load Inmediate, Registers H and L	(H) ← (byte 3)	0	0	1	0	0	0	0	1	3	3	10					
			(L) ← (byte 2)																
LXI	SP	Load Inmediate, Stack Pointer	(SPH) ← (byte 3)	0	0	1	1	0	0	0	1	3	3	10	Queste istruzioni non influenzano lo *stato*				
			(SPL) ← (byte 2)																
MOV	M, r	Move to Memory	((H) (L)) ← (r)	0	1	1	1	0	*S*	*S*	*S*	1	2	7					
MOV	r, M	Move from Memory	(r) ← ((H) (L))	0	1	*D*	*D*	*D*	1	1	0	1	2	7					
MOV	r1, r2	Move Registers	(r1) ← (r2)	0	1	*D*	*D*	*D*	*S*	*S*	*S*	1	1	5					
MVI	M	Move to Memory Inmediate	((H) (L) ← (byte 2)	0	0	1	1	0	1	1	0	2	3	10					
MVI	r	Move inmediate	(r) ← (byte 2)	0	0	*D*	*D*	*D*	1	1	0	2	2	7					
SHLD		Store H and L Direct	((byte 3) (byte 2)) ← (L)	0	0	1	0	0	0	1	0	3	5	16					
			((byte 3) (byte 2)+1)) ← (H)																
STA		Store Accumulator Direct	((byte 3) (byte 2)) ← (A)	0	0	1	1	0	0	1	0	3	4	13					
STAX	B	Store Accumulator Indirect	((B)) ← (A)	0	0	0	0	0	0	1	0	1	2	7					
STAX	D	Store Accumulator Indirect	((D)) ← (A)	0	0	0	1	0	0	1	0	1	2	7					
XCHG		Exchange H and L with D and E	(H) ← (D)	1	1	1	0	1	0	1	1	1	1	4					
			(L) ↔ (E)																

Mnemonico		Descrizione	Operazione	Codice macchina								Num. bytes M	Num. cicli T	Num. MC	Stato				
				7	6	5	4	3	2	1	0				S	Z	AC	P	CY
Istruzioni logiche																			
ANA	M	AND Memory	(A) ← (A) ∧ ((H) (L))	1	0	1	0	0	1	1	0	1	2	7	↕	↕		↕	0
ANA	r	AND Register	(A) ← (A) ∧ (r)	1	0	1	0	0	S	S	S	1	1	4	↕	↕		↕	0
ANI		AND Inmediate	(A) ← (A) ∧ (byte 2)	1	1	1	0	0	1	1	0	2	2	7	↕	↕		↕	0
CMA		Complement Accumulator	(A) ← $(\bar{A})$	0	0	1	0	1	1	1	1	1	1	4					
CMC		Complement Carry	(CY) ← $(\overline{CY})$	0	0	1	1	1	1	1	1	1	1	4					↕
CMP	M	Compare Memory	(A) − ((H) (L))	1	0	1	1	1	1	1	0	1	2	7	↕	↕ [d]	↕	↕	↕ [a]
CMP	r	Compare Register	(A) − (r)	1	0	1	1	1	S	S	S	1	1	4	↕	↕ [b]	↕	↕	↕ [b]
CPI		Compare Inmediate	(A) − (byte 2)	1	1	1	1	1	1	1	0	2	2	7	↕	↑ [c]	↕	↕	↕ [c]
ORA	M	OR Memory	(A) ← (A) v ((H) (L))	1	0	1	1	0	1	1	0	1	2	7	↕	↕	0	↕	0
ORA	r	OR Register	(A) ← (A) v (r)	1	0	1	1	0	S	S	S	1	1	4	↕	↕	0	↕	0
ORI		OR Inmediate	(A) ← (A) v (byte 2)	1	1	1	1	0	1	1	0	2	2	7	↕	↕	0	↕	0
RAL		Rotate Left through Carry	(An+1) ← An); (CY) ← (A_7) (A_0) (CY)	0	0	0	1	0	1	1	1	1	1	4					↕
RAR		Rotate Right through Carry	(An) ← (An+1); (CY) ← (A_0) (A_7) ← (CY)	0	0	0	1	1	1	1	1	1	1	4					↕
RLC		Rotate Left	(An+1) ← (An); (A_0) ← (A_7) (CY) ← (A_7)	0	0	0	0	0	1	1	1	1	1	4					↕
RRC		Rotate Right	(An) ← (A^{+1}); (A_7) ← (A_0) (CY) ← (A_0)	0	0	0	0	1	1	1	1	1	1	4					↕
STC		Set Carry	(CY) ← 1	0	0	1	1	0	1	1	1	1	1	4					1
XRA	M	Exclusive OR Memory	(A) ← (A) ∀ ((H) (L))	1	0	1	0	1	1	1	0	1	2	7	↕	↕	0	↕	0
XRA	r	Exclusive OR Register	(A) ← (A) ∀ (r)	1	0	1	0	1	S	S	S	1	1	4	↕	↕	0	↕	0
XRI		Exclusive OR Inmediate	(A) ← (A) ∀ (byte 2)	1	1	1	0	1	1	1	0	2	2	7	↕	↕	0	↕	0

Mnemonico		Descrizione	Operazione	Codice macchina 7	6	5	4	3	2	1	0	Num. bytes M	Num. cicli T	Num. MC	Stato S	Z	AC	P	CY
Istruzioni aritmetiche																			
ACI		Add Inmediate whit Carry	(A) ← (A)+(byte 2)+(CY)	1	1	0	0	1	1	1	0	2	2	7	↕	↕	↕	↕	↕
ADC	M	Add Memory whit Carry	(A) ← (A)+((H) (L))+(CY)	1	0	0	0	1	1	1	0	1	2	7	↕	↕	↕	↕	↕
ADC	r	Add Register whit Carry	(A) ← (A)+(r)+(CY)	1	0	0	0	1	S	S	S	1	1	4	↕	↕	↕	↕	↕
ADD	M	Add Memory	(A) ← (A)+((H) (L))	1	0	0	0	0	1	1	0	1	2	7	↕	↕	↕	↕	↕
ADD	r	Add Register	(A) ← (A)+(r)	1	0	0	0	0	S	S	S	1	1	4	↕	↕	↕	↕	↕
ADI		Add Inmediate	(A) ← (A)+(byte 2)	1	1	0	0	0	1	1	0	1	2	7	↕	↕	↕	↕	↕
DAA		Decimal Adjust Accumulator	8bit number in Accumulator is converted to two 4 bit BCD digits	0	0	1	0	0	1	1	1	1	1	4	↕	↕	↕	↕	↕
DAD	B	Add B and C to H and L	(H) (L) ← (H) (L)+(B) (C)	0	0	0	0	1	0	0	1	1	3	10					↕
DAD	D	Add D and E to H and L	(H) (L) ← (H) (L)+(D) (E)	0	0	0	1	1	0	0	1	1	3	10					↕
DAD	H	Add H and L to H and L	(H) (L) ← (H) (L)+(H) (L)	0	0	1	0	1	0	0	1	1	3	10					↕
DAD	SP	Add Stack Pointer to H and L	(H) (L) ← (H) (L)+(SP)	0	0	1	1	1	0	0	1	1	3	10					↕
DCR	M	Decrement Memory	((H) (L)) ← ((H) (L))-1	0	0	1	1	0	1	0	1	1	3	10	↕	↕	↕	↕	
DCR	r	Decrement Register	(r) ← (r)-1	0	0	D	D	D	1	0	1	1	1	5	↕	↕	↕	↕	
DCX	B	Decrement Registers B and C	(B) (C) ← (B) (C)-1	0	0	0	0	1	0	1	1	1	1	5					
DCX	D	Decrement Registers D and E	(D) (E) ← (D) (E)-1	0	0	0	1	1	0	1	1	1	1	5					
DCX	H	Decrement Registers H and L	(H) (L) ← (H) (L)-1	0	0	1	0	1	0	1	1	1	1	5					
DCX	SP	Decrement Stack Pointer	(SP) ← (SP)-1	0	0	1	1	1	0	1	1	1	1	5					
INR	M	Increment Memory	((H) (L)) ← ((H) (L))+1	0	0	1	1	0	1	0	0	1	3	10	↕	↕	↕	↕	
INR	r	Increment Register	(r) ← (r)+1	0	0	D	D	D	1	0	0	1	1	5	↕	↕	↕	↕	
INX	B	Increment Registers B and C	(B) (C) ← (B) (C)+1	0	0	0	0	0	0	1	1	1	1	5					
INX	D	Increment Registers D and E	(D) (E) ← (D) (E)+1	0	0	0	1	0	0	1	1	1	1	5					
INX	H	Increment Registers H and L	(H) (L) ← (H) (L)+1	0	0	1	0	0	0	1	1	1	1	5					
INX	SP	Increment Stack Pointer	(SP) ← (SP)+1	0	0	1	1	0	0	1	1	1	1	5					
SBB	M	Subtract Memory with Borrow	(A) ← (A) - ((H) (L))-(CY)	1	0	0	1	1	1	1	0	1	2	7	↕	↕	↕	↕	↕
SBB	r	Subtract Register whit Borrow	(A) ← (A)-(r)-(CY)	1	0	0	1	1	S	S	S	1	1	4	↕	↕	↕	↕	↕
SBI		Subtract Inmediate with Borrow	(A) ← (A)-(byte 2)-(CY)	1	1	0	1	1	1	1	0	2	2	7	↕	↕	↕	↕	↕
SQB	M	Subtract Memory	(A) ← (A)-((H) (L))	1	0	0	1	0	1	1	0	1	2	7	↕	↕	↕	↕	↕
SUB	r	Subtract Register	(A) ← (A)-(r)	1	0	0	1	0	S	S	S	1	1	4	↕	↕	↕	↕	↕
SUI		Subtract Inmediate	(A) ← (A)-(byte 2)	1	1	0	1	0	1	1	0	2	2	7	↕	↕	↕	↕	↕

Mnemonico	Descrizione	Operazione	Codice macchina								Num. bytes M	Num. cicli T	Num. MC	Stato				
			7	6	5	4	3	2	1	0				S	Z	AC	P	CY
Istruzioni di salto																		
CALL	Call Unconditional	((SP)−1) ← (PCH) ((SP−2) ← (PCL) (SP) ← (SP)−2 (PC) ← (byte 3) (byte 2)	1	1	0	0	1	1	0	1	3	5	17	Queste istruzioni non influenzano lo *stato*				
CC	Call on Carry	If CY=1, ((SP)−1) ← (PCH) ((SP)−2) ← (PCL) (SP) ← (SP)−2 (PC) ← (byte 3) (byte 2)	1	1	0	1	1	1	0	0	3	3/5	11/17					
CM	Call on Minus	If S=1, ((SP)−1) ← (PCH) ((SP)−2) ← (PCL) (SP) ← (SP)−2 (PC) ← (byte 3) (byte 2)	1	1	1	1	1	1	0	0	3	3/5	11/17					
CNC	Call on No Carry	If CY=0, ((SP)−1) ← (PCH) ((SP)−2) ← (PCL) (SP) ← (SP)−2 (PC) ← (byte 3) (byte 2)	1	1	0	1	0	1	0	0	3	3/5	11/17					
CNZ	Call on Not Zero	If Z=0, ((SP)−1) ← (PCH) ((SP)−2) ← (PCL) (SP) ← (SP)−2 (PC) ← (byte 3) (byte 2)	1	1	0	0	0	1	0	0	3	3/5	11/17					
CP	Call on Positive	If S=0, ((SP)−1) ← (PCH) ((SP)−2) ← (PCL) (SP) ← (SP)−2 (PC) ← (byte 3) (byte 2)	1	1	1	1	0	1	0	0	3	3/5	11/17					

Mnemonico	Descrizione	Operazione	Codice macchina 7	6	5	4	3	2	1	0	Num. bytes M	Num. cicli T	Num. MC	Stato S	Z	AC	P	CY
CPE	Call on Parity Even	If P=1, ((SP)-1) ← (PCH) ((SP)-2) ← (PCL) (SP) ← (SP)-2 (PC) ← (byte 3) (byte 2)	1	1	1	0	1	1	0	0	3	3/5	11/17					
CPO	Call on Parity Odd	If P=0, ((SP)-1) ← (PCH) ((SP)-2) ← (PCL) (SP) (SP)-2 (PC) ← (byte 3) (byte 2)	1	1	1	0	0	1	0	0	3	3/5	11/17					
CZ	Call on Zero	If Z=1, ((SP)-1) (PCH) ((SP)-2) ← (PCL) (SP) ← (SP)-2 (PC) ← (byte 3) (byte 2)	1	1	0	0	1	1	0	0	3	3/5	11/17					
JC	Jump on Carry	If CY=1, (PC) ← (byte 3) (byte 2)	1	1	0	1	1	0	1	0	3	3	10	Queste istruzioni non influenzano lo *stato*				
JM	Jump on Minus	If S=1, (PC) ← (byte 3) (byte 2)	1	1	1	1	1	0	1	0	3	3	10					
JMP	Jump Unconditional	(PC) ← (byte 3) (byte 2)	1	1	0	0	0	0	1	1	3	3	10					
JNC	Jump on No Carry	If CY=0, (PC) ← (byte 3) (byte 2)	1	1	0	1	0	0	1	0	3	2	10					
JNZ	Jump on Not Zero	If Z=0, (PC) ← (byte 3) (byte 2)	1	1	0	0	0	0	1	0	3	3	10					
JP	Jump on Positive	If S=0, (PC) ← (byte 3) (byte 2)	1	1	1	1	0	0	1	0	3	3	10					
JPE	Jump on Parity Even	If P=1, (PC) ← (byte 3) (byte 2)	1	1	1	0	1	0	1	0	3	3	10					
JPO	Jump on Parity Odd	If P=0, (PC) ← (byte 3) (byte 2)	1	1	1	0	0	0	1	0	3	3	10					
JZ	Jump on Zero	If Z=1, (PC) ← (byte 3) (byte 2)	1	1	0	0	1	0	1	0	3	3	10					

Mnemonico	Descrizione	Operazione	Codice macchina								Num. bytes M	Num. cicli T	Num. MC	Stato				
			7	6	5	4	3	2	1	0				S	Z	AC	P	CY
PCHL	H and L to Program Counter	(PCH) ← (H) (PCL) ← (L)	1	1	1	0	1	0	0	1	1	1	5	Queste istruzioni non influenzano lo *stato*				
RC	Return on Carry	If CY=1, (PCL) ← ((SP)) (PCH) ← ((SP+1) (SP) ← (SP)+2	1	1	0	1	1	0	0	0	1	1/3	5/11					
RET	Return	(PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	0	0	1	0	0	1	1	3	10					
RM	Return on Minus	If S=1, (PCL) ← ((SP)) (PCH) ← ((SP))+1) (SP) ← (SP)+2	1	1	1	1	1	0	0	0	1	1/3	5/11					
RNC	Return on No Carry	If CY=0, (PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	0	1	0	0	0	0	1	1/3	5/11					
RNZ	Return on Not Zero	If Z=0, (PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	0	0	0	0	0	0	1	1/3	5/11					
RP	Return on Positiva	If S=0, (PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	1	1	0	0	0	0	1	1/3	5/11					
RPE	Return on Parity Even	If P=1, (PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	1	0	1	0	0	0	1	1/3	5/11					
RPO	Return on Parity Odd	If P=0, (PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	1	0	0	0	0	0	1	1/3	5/11					

Mnemonico	Descrizione	Operazione	Codice macchina								Num. bytes M	Num. cicli T	Num. MC	Stato				
			7	6	5	4	3	2	1	0				S	Z	AC	P	CY
Istruzioni di salto																		
RST	Restart	((SP)−1) ← (PCH) ((SP)−2) ← (PCL) (SP) ← (SP)−2 (PC) ← 8.(NNN)	1	1	*N*	*N*	*N*	1	1	1	1	3	11	Queste istruzioni non influenzano lo *stato*				
RZ	Return on Zero	If Z=1 (PCL) ← ((SP)) (PCH) ← ((SP)+1) (SP) ← (SP)+2	1	1	0	0	1	0	0	0	1	1/3	5/11					

Mnemonico	Descrizione	Operazione	Codice macchina								Num. bytes M	Num. cicli T	Num. MC	Stato				
			7	6	5	4	3	2	1	0				S	Z	AC	P	CY
Istruzioni di controllo e dello stack																		
DI	Disable Interrupts	The Interrupt system is disabled following the execution of the Di instruction.	1	1	1	1	0	0	1	1	1	1	1					
EI	Enable Interrupts	The interrupt system is enabled following the execution of next instruction.	1	1	1	1	1	0	1	1	1	1	4					
HLT	Halt	Processor is stopped: registers and flags are unaffected.	0	1	1	1	0	1	1	0	1	1	7					
IN	Input	(A) ← (data)	1	1	0	1	1	0	1	1	2	3	10					

Mnemonico	Descrizione	Operazione	Codice macchina								Num. bytes M	Num. cicli T	Num. MC	Stato				
			7	6	5	4	3	2	1	0				S	Z	AC	P	CY
NOP	No Operation	No operation is performed: registers and flags are unaffected.	0	0	0	0	0	0	0	0	1	1	4	.			.	.
OUT	Output	(data) ← (A)	1	1	0	1	0	0	1	1	2	3	10	.	.		.	.
POP B	Pop Registers B and C off Stack	(C) ← ((SP)) (B) ← ((SP)+1) (SP) ← (SP)+2	1	1	0	0	0	0	0	1	1	3	10	.	.		.	.
POP D	Pop Registers D and E off Stack	(D) ← ((SP)) (E) ← ((SP)+1) (SP) ← (SP)+2	1	1	0	1	0	0	0	1	1	3	10	.	.		.	.
POP H	Pop Registers H and L off Stack	(H) ← ((SP)) (L) ← ((SP)+1) (SP) ← (SP)+2	1	1	1	0	0	0	0	1	1	3	10	.	.		.	.
POP PSW	Pop Accumulator and Flags off Stack	(CY) ← $((SP))_0$ (P) ← $((SP))_2$ (AC) ← $((SP))_4$ (Z) ← $((SP))_6$ (S) ← $((SP))_7$ (A) ← ((SP)+1) (SP) ← (SP)+2	1	1	1	1	0	0	0	1	1	3	10					
PUSH B	Push Registers B and C on Stack	((SP)−1) ← (B) ((SP)−2) ← (C) (SP) ← (SP)−2	1	1	0	0	0	1	0	1	1	3	11	.	.		.	
PUSH D	Push Registers D and D on Stack	((SP)−1) ← (D) ((SP)−2) ← (E) (SP) ← (SP)−2	1	1	0	1	0	1	0	1	1	3	11	.	.		.	.
PUSH H	Push Registers H and L on Stack	((SP)−1) ← (H) ((SP)−2) ← (L) (SP) ← (SP)−2	1	1	1	0	0	1	0	1	1	3	11	.	.		.	.

Mnemonico	Descrizione	Operazione	Codice macchina 7	6	5	4	3	2	1	0	Num. bytes M	Num. cicli T	Num. MC	Stato S	Z	AC	P	CY
PUSH PSW	Push Accumulator and Flags on Stack	((SP) - 1) ← (A) $((SP)-2)_0$ ← (CY) $((SP)-2)_1$ ← 1 $((SP)-2)_2$ ← (P) $((SP)-2)_3$ ← 0 $((SP)-2)_4$ ← (AC) $((SP)-2)_5$ ← 0 $((SP)-2)_6$ ← (Z) $((SP)-(SP)_7$ ← (S) (SP) ← (SP) - 2	1	1	1	1	0	1	0	1	1	3	11	.	.		.	.
SPHL	Move H and L to Stack Pointer	(SP) ← (H) (L)	1	1	1	1	1	0	0	1	1	1	5	.	.		.	.
XTHL	Exchange Top of Stack with H and L	(L) ↔ ((SP)) (H) ↔ ((SP) + 1)	1	1	1	0	0	0	1	1	1	5	18	.	.		.	.

NOTE:

a) Z=1 se (A)=(H)(L)
CY=1 se (A)>(H)(L)

b) Z=1 se (A)=(r)
CY=1 se (A)>(r)

c) Z=1 se (A)=(byte 2)
CY=1 se (A)>(byte)

SIMBOLI E ABBREVIAZIONI

I seguenti simboli e abbreviazioni sono stati usati nella descrizione delle istruzioni dell'8080. A=Registro A (Accumulatore). B=Registro B. C=Registro C. D=Registro D. E=Registro E. H=Registro H. L=Registro L. DDD,SSS=Il bit padrone designa uno dei registri A, B, C, D, E, H, L. (DDD=Destinazione, SSS=Sorgente):

DDD o SSS	NOME DEL REG.
111	A
000	B
001	C
010	D
011	E
100	H
101	L

Byte 2: Il secondo byte dell'istruzione.
Byte 3: Il terzo byte dell'istruzione.
Riporto: 8-bits di indirizzo per un circuito di ingresso/uscita.
r, r1, r2: Uno dei registri A, B, C, D, E, H, L.

SIMBOLI E SIGNIFICATO

PC = Registro contatore di programma a 16 bit (PCH e PCL), suddiviso in due gruppi da 8 bits. SP = Registro puntatore di Stack a 16 bits (SPH e SPL), suddiviso in due gruppi da 8 bits. ( ) = Il contenuto della memoria o del registro si racchiude tra parentesi. ← = E' sostituito da. ∧ = Logica AND. ⊻ = OR Esclusivo. ∨ = OR inclusivo. + = Somma. - = Sottrazione, complemento a due. * = Moltiplicazione. ←→ = Scambio. ‾ = Complemento, esempio (A). n = numeri da 0 a 7. NNN = Rappresentazione in binario da 000 a 111, da 0 a 7, rispettivamente. • = Non influenzato. 0 = Reset. 1 = Set. x = Indifferente. ↕ = Flags influenzati.

Tabella II.-Tabella delle istruzioni del μP 8080.

- le istruzioni di differenza SUBTRACT, che effettuano una sottrazione tra registri, con o senza *borrow* (cioè prestito) e tra questi e la memoria.

- Nel gruppo delle istruzioni logiche esistono:

- le istruzioni che eseguono AND, OR, e OR ESCLUSIVO, tra la memoria e l'accumulatore, o tra questo e i registri; l'istruzione di complemento esegue un'inversione del contenuto dell'accumulatore;
- le istruzioni di confronto che effettuano un confronto logico bit a bit tra l'accumulatore e i registri; le istruzioni di rotazione ROTATE, che eseguono una rotazione, verso destra o sinistra, nei diversi registri, con o senza carry.

- Nel gruppo delle istruzioni di salto o ramificazione si trovano le istruzioni di salto JUMP con o senza condizione positiva, negativa, di zero, ecc., in base al contenuto dei bit dello status.

- Le istruzioni di ritorno RETURN, che servono per uscire da una subroutine, con o senza condizione di zero, negativa, positiva, ecc., secondo lo status.

- Tra le miscellanee vi sono:

- le istruzioni HALT e NOP;
- le istruzioni di uso dello STACK, che caricano il contenuto dei vari registri nello STACK.

Tabella delle istruzioni in funzione del codice macchina

Nella tabella III sono contenute tutte le istruzioni già note, ma in base al loro codice macchina, per cui un'istruzione si cerca partendo dal suo codice esadecimale, in modo che il primo digit corrisponda alla colonna (I) e il secondo alla riga (J), nel punto di intersezione tra riga e colonna esiste l'istruzione cercata.

Questo metodo di ricerca di un'istruzione viene utilizzato per disassemblare pazientemente un programma di cui si conosce il codice macchina.

Il microprocessore 8085

Questo microprocessore fu commercializzato dall'Intel dopo l'8080. Il µP 8085 ha quindi beneficiato di tutta l'evoluzione tecnica degli ultimi anni,

I \ J	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B	STAX B	INX B	INR B	DCR B	MVI B	RLC		DAD B	LDAX B	DCX B	INR C	DCR C	MVI C	RRC
1		LXI D	STAX D	INX D	INR D	DCR D	MVI D	RAL		DAD D	LDAX D	CDX D	INR E	DCR E	MVI E	RAR
2		LXI H	SHLD adr	INX H	INR H	DCR H	MVI H	DAA		DAD H	LHLD adr	DCX H	INR L	DCR L	MVI L	CMA
3		LXI SP	STA adr	INX SP	INR M	DCR M	MVI M	STC		DAD SP	LDA adr	DCX SP	INR A	DCR A	MVI A	CMC
4	MOV BB	MOV BC	MOV BD	MOV BE	MOV BH	MOV BL	MOBB M	MOV BA	MOV CB	MOV CC	MOV CD	MOV CE	MOV CH	MOV CL	MOV CM	MOV CA
5	MOV DB	MOV DC	MOV DD	MOV DE	MOV DH	MOV DL	MOV DM	MOV DA	MOV EB	MOV EC	MOV ED	MOV EE	MOV EH	MOV EL	MOV EM	MOV EA
6	MOV HB	MOV HC	MOV HD	MOV HE	MOV HH	MOV HL	MOV HM	MOV HA	MOV LB	MOV LC	MOV LD	MOV LE	MOV LH	MOV LL	MOV LM	MOV LA
7	MOV MB	MOV MC	MOV MD	MOV ME	MOV MH	MOV ML	HLT	MOV MA	MOV AB	MOV AC	MOV AD	MOV AE	MOV AH	MOV AL	MOV AM	MOV AA
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB F	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA C	XRA D	XRA E	XRA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ adr	JMP adr	CNZ B	PUSH	ADI O	RST	RZ	RET	JZ adr		CZ adr	CALL adr	ACI	RST 1
D	RNC	POP D	JNC adr	OUT adr	CNC adr	PUSH D	SUI	RST 2	RC		JC adr	IN adr	CC adr		SBI	RST 3
E	RPO	POP H	JPO adr	NTHL	CPO adr	PUSH H	ANI	RST 4	RPE		PCHL	JPE adr	XCHG	CPE adr	XRI	RST 5
F	RP	POP	JP	DI	CP	PUSH	ORI	RST	RM	SPHI	JM adr	EI	CM adr		CPI	RST 4

Tabella III.-Tabella delle istruzioni dell'8080, in base al codice esadecimale.
Esempio: I=C, J=3, istruzione C3=JMP adr (istruzione di salto).

nel campo dell'integrazione. L'8085 è completamente compatibile, a livello logico, con l'8080, e molto migliorato per quanto riguarda gli interrupts. L'8085 contiene la circuiteria ausiliaria che era necessaria all'8080, come i circuiti integrati 8224 e 8228, per cui è un microprocessore molto utilizzato, e gode di grande popolarità nei circuiti tecnici professionali.

Caratteristiche dell'8085

Il μP 8085 possiede le seguenti caratteristiche tecniche e funzionali:

- è realizzato su un solo CHIP in tecnologia NMOS;
- ha una capacità di indirizzamento di 64 Kbytes;
- il periodo di clock è di 1,3 μsec nell'8085 A, e di 0,8 μsec nella versione A-2;
- prevede un'unica alimentazione a 5 V;
- dispone di 78 istruzioni (quattro in più rispetto all'8080);
- prevede gli indirizzamenti: diretto esteso, immediato, implicito, diretto e indiretto tramite registro;
- il sistema di interrupt dispone di un interrupt non mascherabile (TRAP), e altri livelli di interrupts vettorializzati e mascherabili.

Configurazione interna dell'8085

La configurazione interna è quella classica di tutti i microprocessori, come mostrato in Fig. 11.

In questo microprocessore si distinguono le seguenti unità funzionali:

- il blocco dei registri;
- l'unità di controllo generale;
- l'unità di controllo degli interrupts;
- l'unità di controllo di input/output seriale;
- i registri sono i seguenti:
  - i registri di uso generale sono sei, ad essi ci si può riferire mediante le istruzioni del programma. Possono essere utilizzati come registri da otto bit, o come tre registri da sedici bit ciascuno in funzione del tipo di istruzione da eseguire (ogni istruzione utilizza uno o l'altro metodo). Tali registri si chiamano B, C, D, E, H, L. Per formare registri a sedici bit, si

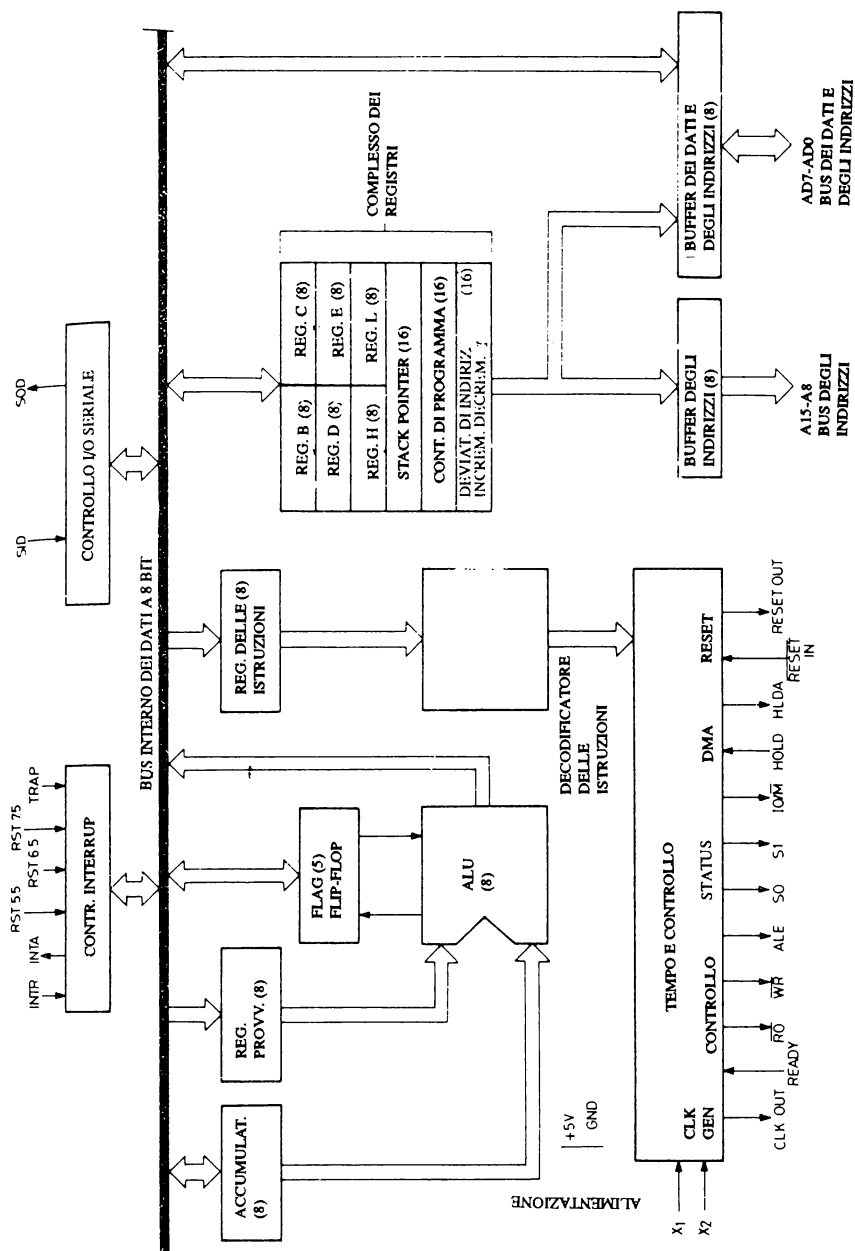



Fig. 11.-Configurazione interna del microprocessore 8085.

raggruppano a due a due nel modo seguente: B con C, D con E, e H con L.

- il contatore di programma (PC) è un registro a sedici bit, contenente l'indirizzo in cui si trova la successiva istruzione da eseguire che si autoincrementa ad ogni ciclo di lavoro.
- il puntatore di stack (SP), registro a 16 bit che contiene l'indirizzo dell'ultima posizione occupata dello stack. Lo stack pointer è caricato con l'indirizzo dello stack, per il quale si può utilizzare qualsiasi posizione della memoria; il suo contenuto viene decrementato ogni volta che un indirizzo è memorizzato nello stack, e incrementato ogni volta che un indirizzo viene prelevato dallo stesso stack.
- registri temporanei a otto bit ad uso interno del µP, e quindi inaccessibili al programmatore.
- registro accumulatore a otto bit, detto A.
- registro di stato (status) a otto bit, dei quali solo cinque

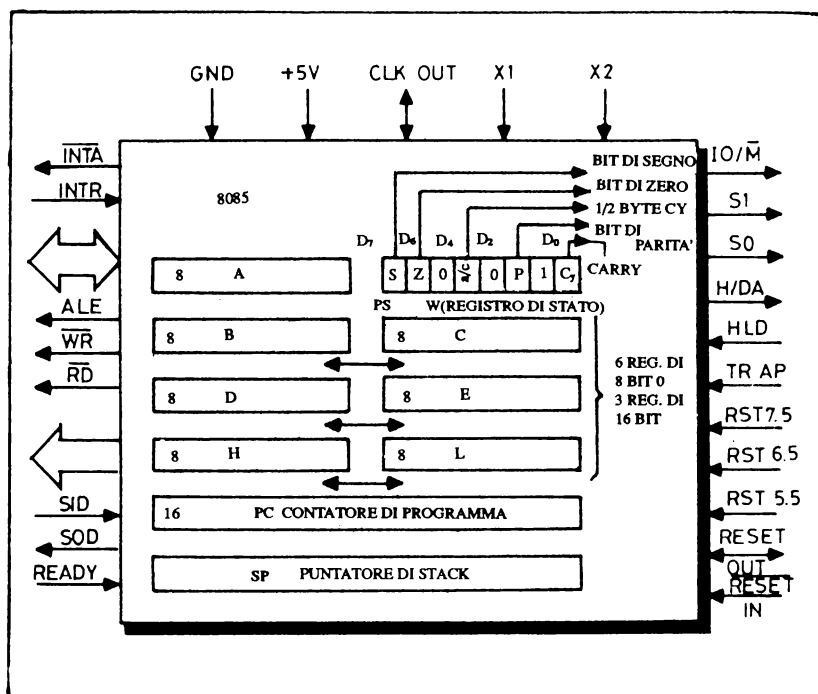



Fig. 12.-Registri interni del µP 8085.

vengono utilizzati con il seguente significato:
D0=Cy, è l'indicatore di carry, utilizzato nelle operazioni aritmetiche; se esiste riporto vale uno, altrimenti zero. Solo se l'operazione è una sottrazione il carry funziona inversamente, e viene resettato a zero se esiste riporto.
D2=P, è l'indicatore di parità, settato a uno se la parità (numero di bit a uno) dell'accumulatore è pari, altrimenti vale zero.
D4=AC (Auxiliar Carry), questo indicatore è settato a uno se esiste un riporto sul bit quattro del dato elaborato, altrimenti a zero.
D6=Z (Zero), indicatore che vale uno se il risultato di un'operazione vale zero, se tale risultato è diverso da zero, viene resettato a zero.
D7=S, indicatore che è riferito al segno del dato nell'accumulatore; se vale uno, il dato è negativo, se zero, il dato è positivo.

Unità di controllo dell'8085

Questo blocco interno del microprocessore è costituito dai seguenti elementi:

- il registro di istruzione da otto bit a cui viene inviato il primo ottetto o byte dell'istruzione, che rappresenta il codice dell'operazione;
- il decodificatore di istruzioni, che preleva il contenuto del registro di istruzione per determinare l'operazione da effettuare; l'uscita del decodificatore controlla tutti i registri, l'ALU, e i bus dei dati e degli indirizzi;
- il sequenziatore, che genera le microistruzioni di lettura o scrittura, in funzione dell'informazione fornita dal decodificatore;
- il generatore dei segnali interni di clock, incorporato nel µP 8085, che richiede solo l'utilizzo di un cristallo di quarzo per stabilire la sequenza delle operazioni. La circuiteria interna del clock genera due segnali Φ_1 e Φ_2 che attivano il funzionamento del microprocessore, e non sono disponibili all'esterno.

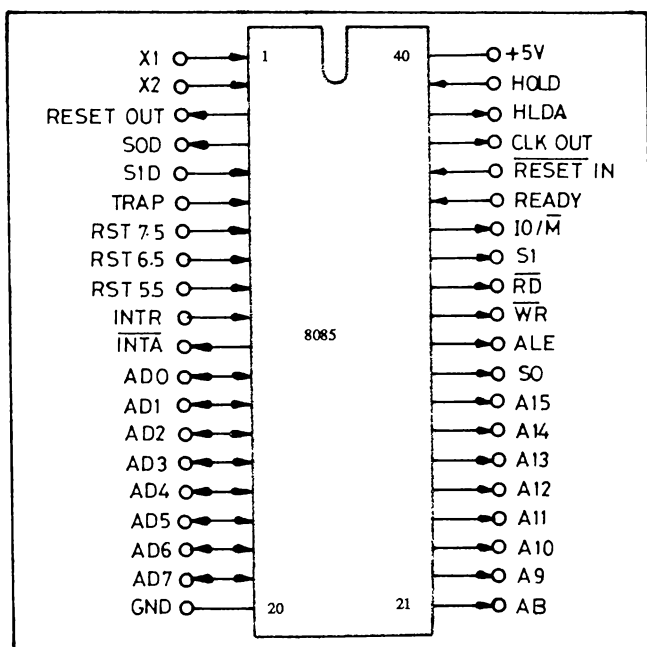



Fig. 13.-Distribuzione delle funzioni sui terminali del μP 8085.

Nomenclatura dei terminali dell'8085

Anche il microprocessore 8085 è contenuto nel classico contenitore a 40 piedini. In Fig. 13 si può rilevare la corrispondenza tra la funzione e il numero del terminale. Le funzioni sono le seguenti:

- A8-A15, è il bus degli indirizzi: dei 16 bit che compongono il bus per indirizzare il campo di 64 Kbytes, solo otto escono verso l'esterno, gli altri da A0 ad A7 escono al momento opportuno dal bus dei dati.
- AD0-AD7, questi terminali costituiscono il bus dei dati attraverso cui circola il flusso di dati, da e verso l'esterno, e la parte bassa dell'indirizzo.
- ALE (ADDRESS LATCH ENABLE) è il segnale che consente di caricare in LATCH la parte bassa del bus degli indirizzi per completare l'indirizzo da A0 ad A15. Questo segnale esce nel primo ciclo di ogni istruzione.

IO/M	*So*	*S1*	*Ciclo del microprocessore*
0	0	1	MW (scrittura in memoria)
0	1	0	MR (lettura in memoria)
1	0	1	IOW (scrittura periferica IO)
1	1	0	IOR (lettura periferica IO)
0	1	1	(Ricerca del codice operativo)
1	1	1	(Riconoscimento di interrupt)

Tabella IV.- Relazione tra lo stato di alcuni segnali di controllo e il ciclo del microprocessore.

- S0, S1, IO/M, questi segnali definiscono il ciclo di lavoro in cui i trova il μP in ogni istante, in base alla tabella IV.
- RD (READ CONTROL), quando questo segnale è zero, indica alla memoria e alla periferia che il μP esegue un'operazione di lettura.
- WR è simile al precedente, ma indica un'operazione di scrittura.
- READY, è un'ingresso mediante il quale le periferiche indicano al μP che sono pronte per mettersi in contatto con esso, settandolo a uno.
- HOLD è un ingresso attraverso il quale le periferiche o memorie lente occupano il microprocessore, per un certo tempo, mentre questo terminale è a uno logico.
- HLDA (HOLD ACKNOWLEDGE), riconoscimento di occupazione, è un'uscita tramite cui il μP comunica alle periferiche in risposta ad un segnale di HOLD, settandola a uno per tutto il tempo in cui permane il segnale HOLD.
- INTR (INTERRUPT REQUEST) è un ingresso tramite cui le periferiche possono sollecitare un interrupt portandolo a uno.
- INTA (INTERRUPT ACKNOWLEDGE), riconoscimento dell'interrupt, il μP porta a zero quest'uscita quando accetta un interrupt richiesto da una periferica.
- RST5.5, RST6.5, e RST7.5 (RESTART INTERRUPT) sono segnali di richiesta di interrupt dello stesso tipo di INTA, con la differenza che questi tre provocano una reinizializzazione interna del μP, posizionandolo a locazioni fisse di memoria.
- TRAP è un altro ingresso di richiesta di interrupt, ma con priorità su tutte le altre.
- RESET IN, ingresso di azzeramento per l'inizializzazione del μP

dopo aver fornito la tensione, che lo posiziona all'indirizzo 0000.

- RESET OUT, questa uscita passa a uno durante l'inizializzazione del µP causata dal segnale precedente.
- CLK è un'uscita del clock del µP, e può servire come clock di sistema.
- X1, X2 sono due ingressi a cui si collega il cristallo di quarzo.
- SID (SERIAL INPUT DATA) è un ingresso seriale. Il contenuto di questo bit compare nel bit 7 dell'accumulatore, quando si esegue la corrispondente istruzione.
- SOD (SERIAL OUTPUT DATA) è un'uscita di dati seriali, il dato che esce è il contenuto del bit 7 dell'accumulatore.
- V_{CC} e V_{SS} sono i terminali di alimentazione, +5 V, e massa, rispettivamente.

Ottenimento del bus degli indirizzi

Il µP dispone di 16 bit per il bus degli indirizzi, dei quali gli otto più significativi (A8-A15) escono direttamente dal µP verso l'esterno, gli altri otto meno significativi (A0-A7) escono con il bus dei dati durante un ciclo previ-

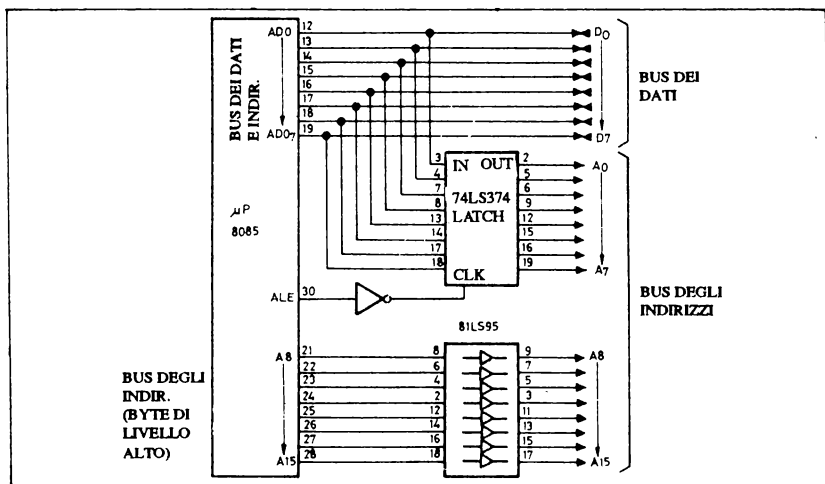



Fig. 14.-Dettaglio dell'ampliamento del bus degli indirizzi.

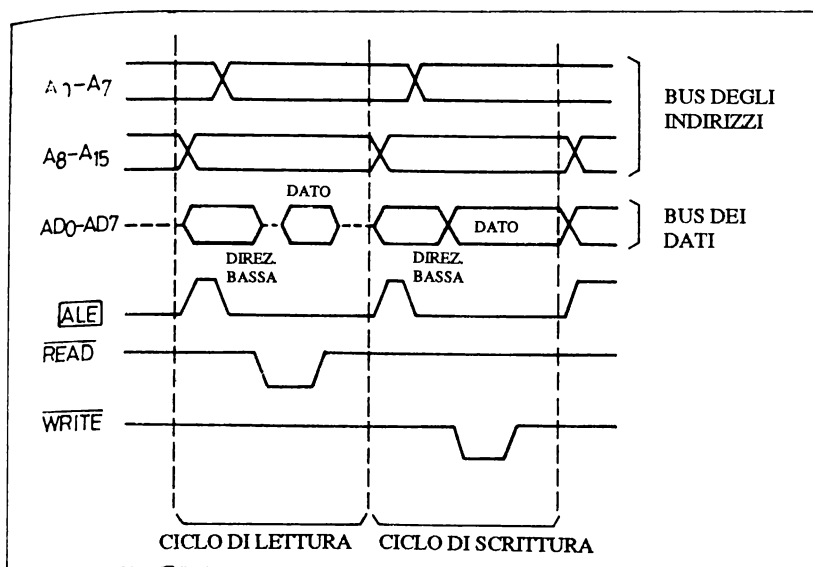



Fig. 15.-Diagramma dei tempi relativo al segnale ALE per ottenere il bus dai dati completo.

sto per l'esecuzione dell'istruzione; vale a dire che dal bus dei dati esce dapprima la parte bassa del bus degli indirizzi e poi il codice dell'istruzione. Questa tecnica permette di ridurre il numero dei terminali del µP, a soli quaranta.

Il segnale ALE (ADDRESS LATCH ENABLE) è quello che coordina la trasformazione del bus dei dati, in indirizzo o dato. Quando compare tale segnale nel primo ciclo di ogni istruzione che accede alla memoria, si caricano i latch con il contenuto del bus dei dati; presentandolo all'uscita, questo contenuto definisce gli otto bit meno significativi del bus degli indirizzi. Il segnale ALE attiva il clock dei latch per caricarli.

Nel diagramma dei tempi di Fig. 15, si possono rilevare i principali segnali del µP correlati con il segnale ALE.

Uso degli ingressi ready e hold

Quando si porta a livello logico zero l'ingresso READY del µP, si genera lo stato di attesa (WAIT). Mediante questo stato il µP può accedere alle memorie o alle periferiche lente. Tramite questo ingresso, si può anche ottenere

l'esecuzione passo a passo di un programma. Se, tramite una circuiteria esterna ausiliaria, si resetta a zero questo terminale durante l'esecuzione di un'istruzione, i bus mantengono l'indirizzo e il dato, per cui l'utente può accertare se il programma viene correttamente eseguito. Da quanto detto, risulta che questo è un metodo lento, ma efficace, per il controllo di un programma.

Il segnale HOLD può essere utilizzato per implementare (realizzare) un sistema DMA (Direct Memory Access). Esistono alcune periferiche che, per funzionare più velocemente, necessitano di accedere direttamente alla memoria senza passare attraverso la CPU; per ottenere ciò è necessario che il µP si arresti e ponga i bus ad alta impedenza, per permettere in tal modo che la suddetta periferica possa accedere alla stessa memoria utilizzata dal µP per leggere o scrivere un blocco di dati. Con il µP 8085 ciò si ottiene utilizzando l'ingresso HOLD; portando tale ingresso a uno, il µP completa l'esecuzione dell'istruzione in corso, e dopo aver settato a uno l'uscita HLDA, setta i bus ad alta impedenza. Quando la periferica termina di accedere alla memoria, resetta nuovamente a zero l'ingresso HOLD, per cui il microprocessore continuerà con l'istruzione successiva.

Esempio di collegamento del µP 8085

E' simile a quello del µP 8080, ma più semplice e completo, poichè essendo più integrato, i collegamenti esterni sono semplificati. Nell'esempio proposto si osservano tre blocchi, corrispondenti a quelli a cui il µP dovrebbe essere collegato, ma il microprocessore si potrebbe collegare ad altri blocchi o periferiche in funzione dell'utilizzo a cui viene destinato. Bisogna però ricordare di posizionare la PROM, in cui risiedono le istruzioni iniziali, all'indirizzo 0000, in quanto il µP, dopo il reset, riparte da tale indirizzo.

Il blocco definito decodificatore di pagina è, in realtà, quello che si incarica di abilitare i blocchi corrispondenti, in funzione dei bit A13, A14 e A15 del bus diretto degli indirizzi, e il segnale di permesso IO/M che consente il passaggio sincronizzato al decodificatore.

Si osservi il segnale ALE che carica il latch con il contenuto del bus dei dati, per formare la parte bassa (L) del bus degli indirizzi. I segnali alla sinistra del µP sono segnali di interrupts, di cui si tratterà nel seguito, e gli altri sono segnali di controllo già visti.

Gli interrupts nell'8085

Il microprocessore 8085 possiede un complesso e completo sistema di

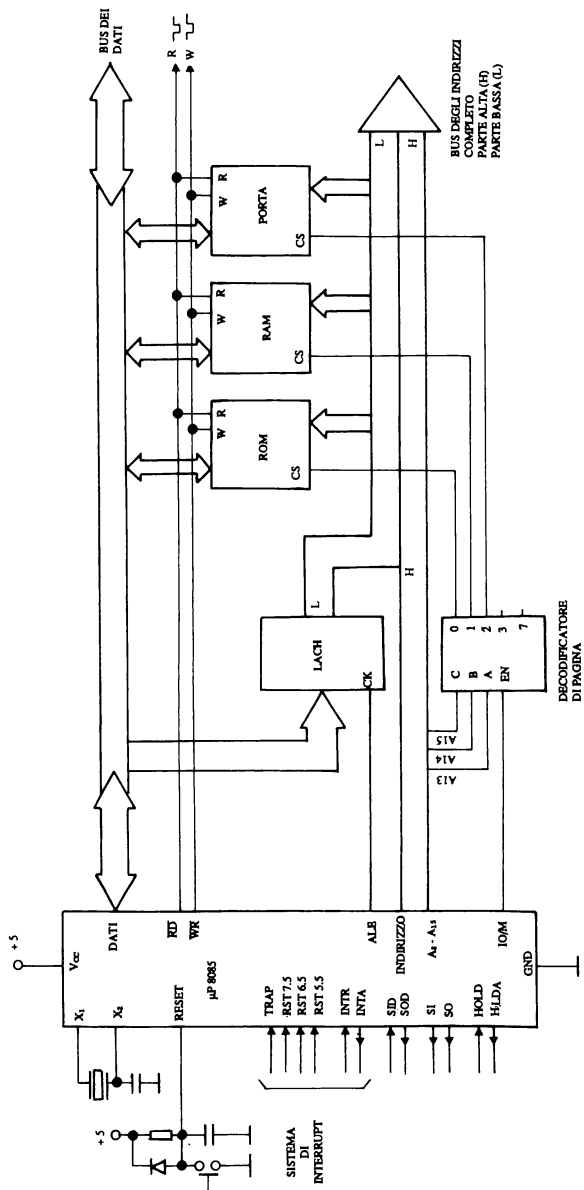



Fig. 16.-Schema dei collegamenti dell'8085.

interrupts, ai quali sono dedicati i chiarimenti seguenti. Il µP 8085 possiede cinque terminali destinati al trattamento degli interrupts.

Si ricorda che un interrupt è un artificio hardware/software, mediante il quale è possibile arrestare il programma in corso in modo che, quando avviene un determinato evento, il µP completi l'istruzione in corso ed effettui un salto ad una determinata subroutine, durante la quale si manterrà l'interrupt, dopo di che, il µP continuerà con l'istruzione successiva del programma principale.

L'8085 dispone di tre diversi modi per trattare gli interrupts, che gli arrivano attraverso i cinque terminali, i cui nomi sono:

- INTR (Interrupt Request): tramite questo ingresso si invia un interrupt che viene accettato o meno in base a quanto è stato indicato in precedenza mediante le istruzioni EI (interrupt permesso) o DI (interrupt non permesso); ciò significa che l'interrupt INTR è mascherabile con tali istruzioni, in modo tale che se nell'esecuzione di un programma è stata inclusa l'istruzione DI, anche se viene richiesto un interrupt tramite il terminale INTR, questo non sarà mai soddisfatto. Questo terminale INTR determina perciò solo uno dei tre modi di trattare un interrupt. Quando un interrupt è permesso e si è prodotto, la CPU cerca un'istruzione di reinizializzazione RST (a un solo byte), che viene fornita dall'esterno tramite la periferica che ha generato l'interrupt. Questo byte ha la forma di Fig. 17, in cui i bit indicati con XXX possono assumere i valori da 000 a 111; ciascuno di questi valori indica in quale degli otto possibili indirizzi, rappresentati anch'essi in Fig. 17, si trova la subroutine che elabora l'interrupt; al termine di tale subroutine deve esistere un'istruzione di ritorno per far sì che il programma, dopo averla eseguita, ritorni a quello principale.
- RST 5.5, RST 6.5, e RST 7.5: la funzione di questi tre terminali è di produrre un secondo modo di trattamento degli interrupts. I terminali RST 5.5 e RST 6.5 rilevano un interrupt solo se il segnale che viene loro applicato è ad uno logico o livello alto, per un certo tempo, come per il precedente ingresso INTR; l'ingresso corrispondente al terminale RST 7.5 è invece eccitato dal fronte di una variazione di segnale da zero a uno, fronte che cambia stato a un bistabile interno al µP. Terminato il trattamento dell'interrupt il µP resetta nuovamente tale bistabile.

  Ciascuno di questi tre terminali, quando viene attivato, indica una richiesta di interrupt, e ad ognuno corrisponde un unico indirizzo a cui si trova la subroutine che tratta tali richieste, indirizzi visibili

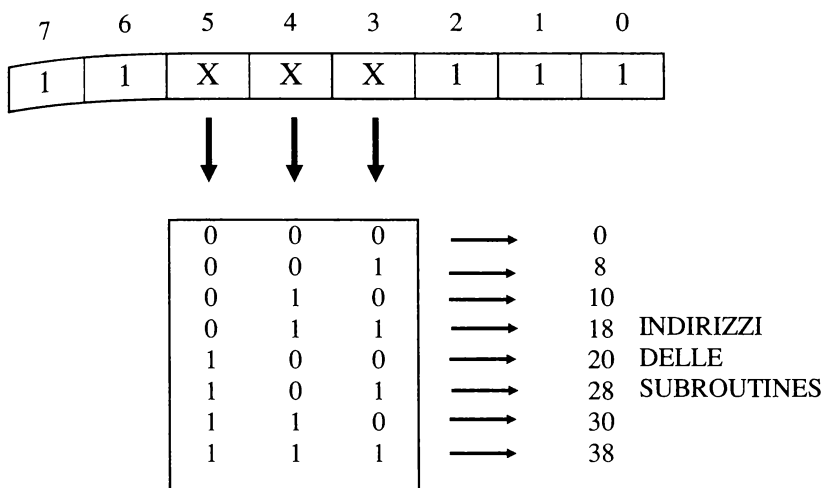



Fig. 17.-Significato dei bit del byte inviato dalla periferica che richiede un interrupt al terminale INTR.

nella tabella V.

Gli interrupts corrispondenti a RST 5.5, RST 6.5 e RST 7.5 possono essere permessi o vietati mediante le solite istruzioni EI e DI, ma sono anche mascherabili via software tramite l'istruzione SIM (Set Interrupt Mask) che permette o meno gli interrupts attraverso la scelta di alcuni bit indicatori di maschera, presenti nel registro accumulatore, che è stato caricato con il contenuto del secondo byte dell'istruzione che funziona da maschera. Dopo l'esecuzione di tale istruzione, saranno permessi solo gli interrupts il cui corrispondente bit sia stato settato a uno logico; ma la maschera potrà essere caricata solo se il bit tre dell'accumulatore è stato settato a uno dal μP. Il bit quattro resetta il bistabile dell'interrupt 7.5.

Mediante l'istruzione RIM (Read Interrupt Mask) è possibile leggere in ogni istante lo stato della maschera con l'informazione complementare dei bit 4, 5, e 6, che indicano gli interrupts corrispondenti che sono stati prodotti, ma che non sono stati accettati fino al momento dell'istruzione RIM.

L'interrupt TRAP è un interrupt non mascherabile, che viene attivato quando il terminale con tale denominazione (TRAP) viene settato a livello logico uno.

Quando il microprocessore lo esegue, questo indirizza una locazione fissa, la 24 esadecimale, a cui deve iniziare la subroutine che processa tale interrupt. Il segnale di interrupt proveniente dall'esterno, che viene fornito a tale terminale, agisce con il fronte di salita sui circuiti interni di interrupt via hardware, ma il livello logico uno deve rimanere finchè non avviene il completo riconoscimento da parte del microprocessore; questo interrupt è individuato dal fronte e dal livello.

Tale interrupt ha la priorità più elevata, per cui può essere utilizzato per trattare i fenomeni più importanti, come ad esempio gli errori, le cadute di alimentazione, ecc.

Nella tabella V sono elencati i vari interrupts, in base al livello di priorità.

Questi livelli sono quelli che determinano l'ordine con cui verranno eseguiti gli interrupts che siano richiesti simultaneamente.

Se durante un programma il microprocessore esegue uno dei cinque interrupt, a partire da tale istante bloccherà il permesso di eseguirne altri finchè non si rimuova tale blocco, via software, mediante l'istruzione EI (permesso di interrupt); tuttavia ciò non avviene per l'interrupt TRAP che, avendo priorità assoluta, può essere trattato in qualsiasi istante. L'esecuzione dell'istruzione RIM dopo che è avvenuto un interrupt TRAP, mostra il dato della maschera degli interrupts ricevuti precedentemente al TRAP, che non sono stati eseguiti.

Livello di priorità	Nome dell'interrupt	Indirizzo della subroutine in esadecimale
Maggior priorità	TRAP	24
–	RST 5.5	2C
–	RST 6.5	34
–	RST 7.5	3C
Minor priorità	INTR	–

Tabella V.-Tabella degli interrupts del microprocessore 8085.

Controllo dell'input/output seriale

Le istruzioni RIM e SIM controllano anche il sistema di input/output seriale che si ottiene tramite il bit 7 dell'accumulatore.

Come è noto, quando il µP esegue un'istruzione RIM, il dato del terminale numero cinque del µP, detto SID, viene caricato nel bit 7 del registro accumulatore, mentre dal terminale numero quattro (SOD) uscirà il contenuto del bit 7 dell'accumulatore, quando viene eseguita l'istruzione SIM, ma solo se il bit 6 dello stesso è stato precedentemente settato a uno.

Il dato all'uscita del terminale SOD verrà mantenuto per un periodo indefinito, poichè tra l'accumulatore e l'uscita esiste un latch in grado di memorizzare tale bit.

In tutti i microprocessori, il sistema di input/output seriale ha una certa importanza, dato che esistono periferiche che comunicano serialmente con la CPU.

L'ingresso SID può essere considerato come un ingresso asincrono attraverso il quale si invia un bit di informazione proveniente dall'esterno, senza che sia necessario stabilire un indirizzo per lo stesso. Tale tipo di ingresso viene anche detto SENSE.

L'uscita SOD può essere considerata come un FLAG. Si ricorda che un FLAG è un segnale a un bit che, se previsto nel programma, può avere un'infinità di applicazioni, come attivare qualche circuito esterno in modo asincrono, o semplicemente attivare un diodo led per indicare se il programma ha percorso una certa subroutine, ecc.

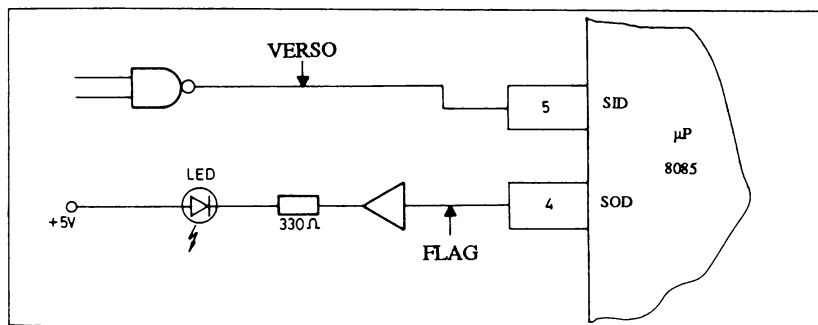



Fig. 18.-Ingresso SID considerato come SENSE e uscita SOD considerata come FLAG.

Simbologia delle istruzioni nell'8085

Nelle tabelle delle istruzioni dell'8085 sono stati utilizzate le sigle e i simboli seguenti:

- r1, r2, e r3 rappresentano alcuni dei registri A, B, C, D, E, H, L e M; l'indirizzo di memoria è determinato mediante H (byte alto) e L (byte basso);
- DDD rappresenta il registro destinazione in cui viene caricato il dato, e SSS il registro sorgente dal quale il dato proviene;
- rp rappresenta una coppia di registri;
- i codici B(B-C): 0, D(D-E): 1, H(H-L): 2, e SP (Stack Pointer): 3, sono i valori I, in esadecimale, della tabella VII;
- n rappresenta un numero compreso tra 0 e 7, e indica una delle otto posizioni a cui salta il contatore di programma dopo un'istruzione RST;
- il codice NNN rappresenta un binario tra 000 e 111;
- CCC indica una condizione dipendente dallo stato dei flags;
- Adr indica un indirizzo.

Modi di indirizzamento dell'8085

Il microprocessore 8085 dispone di vari sistemi di indirizzamento, che ampliano la funzione di ogni istruzione.

Le istruzioni possono essere a uno, due e tre bytes, in base al tipo specifico.

- INDIRIZZAMENTO IMMEDIATO: il byte che segue il codice operativo e il dato o operando, sono istruzioni a due bytes;
- INDIRIZZAMENTO DIRETTO: i due bytes che seguono il codice dell'operazione rappresentano l'indirizzo di memoria in cui si trova il dato;
- INDIRIZZAMENTO TRAMITE REGISTRO: uno dei registri B, C, E, H o L contiene il dato (in questo indirizzamento i registri sono considerati a otto bit);
- INDIRIZZAMENTO TRAMITE COPPIA DI REGISTRI: in questo caso il contenuto della coppia di registri indica l'indirizzo di memoria in cui si trova il dato; i registri sono accoppiati nel seguente modo: B con C, D con E, H con L.

Conoscere il banco dei registri dell'8085 è indispensabile per poterlo programmare, tenendo conto che un registro può essere utilizzato per una istruzione o semplicemente come luogo di immagazzinamento temporaneo di un dato o di un indirizzo.

TABELLA VI

Mnemonico	*Descrizione*	*1° byte dell'istruzione*							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MOV r_1, r_2	$r_1 \leftarrow r_2$ L'informazione contenuta in r_2 passa in r_1	0	1	D	D	D	S	S	S
MOV r_1 dato	$r_1 \leftarrow$ dato Il dato passa in r_1	0	0	D	D	D	1	1	0
LXI rp dato	rp $\leftarrow$ dato Il dato passa nella coppia di registri rp	0	0	R	P	0	0	0	1
LDA mem.	A $\leftarrow$ mem Il contenuto della locazione di memoria indirizzata si trasferisce nell'accumulatore	0	0	1	1	1	0	1	0
STA mem.	mem $\leftarrow$ A Il contenuto dell'accumulatore viene posto nella locazione di memoria indirizzata	0	0	1	1	0	0	1	0
LHLD mem.	HL $\leftarrow$ mem Il contenuto che esiste in una locazione di memoria viene posto nel registro L. Il contenuto dell'indirizzo successivo nel registro H	0	0	1	0	1	0	1	0

Mnemonico	*Descrizione*	*1° byte dell'istruzione*							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SHLD mem.	mem ← HL Trasferisce il contenuto dei registri HL ad una locazione di memoria e alla successiva	0	0	1	0	0	0	1	0
LDAX rp	A ← rp Il contenuto della posizione di memoria indirizzata dalla coppia di registri rp passa nell'accumulatore	0	0	R	P	1	0	1	0
STAX rp	rp ← A Il contenuto dell'accumulatore viene sal vato nella locazione di memoria indirizzata da una coppia di registri	0	0	R	P	0	0	1	0
XCHG	H ↔ D L ↔ E Il contenuto dei due registri H e L si scambia con il contenuto dei registri D e E rispettivamente	1	1	1	0	1	0	1	1
ADD, r	A ← A+r Aggiunge il contenuto del registro r all'accumulatore	1	0	0	0	0	S	S	S
ADI, dato	A ← A+dato Il dato o 2° byte viene sommato all'accumulatore	1	1	0	0	0	1	1	0
ADC, r	A ← A+r+CY Somma il contenuto del registro r e il riporto del CY all'accumulatore	1	0	0	0	1	S	S	S
ACI dato	A ← A+dato+CY Il dato e il riporto vengono sommati all'accumulatore	1	1	0	0	1	1	1	0
SUB r	A ← A-r Sottrae il contenuto del registro r dall'accumulatore	1	0	0	1	0	S	S	S
SUI dato	A ← A-dato Sottrae dato da accumul.re	1	1	0	1	0	1	1	0

Mnemonico	*Descrizione*	*1° byte dell'istruzione*							
		D7	D6	D5	D4	D3	D2	D1	D0
SBBR	A ← A-r-CY Sottrae dall'accumulatore il contenuto del registro r e il riporto	1	0	0	1	1	S	S	S
SBI dato	A ← A-dato-CY Il dato e il riporto vengono sottratti dall'accumulatore	1	1	0	1	1	1	1	0
INR r	r ← r+1 Incrementa contenuto di r	0	0	D	D	D	1	0	0
DCR r	r ← r-1 Decrementa il contenuto del registro r	0	0	D	D	D	1	0	1
INX rp	rp ← rp+1 Il contenuto della coppia di registri rp viene incrementato di un'unità	0	0	R	P	0	0	1	1
DCX rp	rp ← rp-1 Il contenuto della coppia di registri rp viene decrementato di un'unità	0	0	R	P	1	0	1	1
DAD rp	HL ← rp+HL Il contenuto della coppia di registri rp viene sommato al contenuto della coppia di registri H e L.	0	0	R	P	1	0	0	1
DAA	Regolazione decimale nell'accumulatore per ottenere 2 digit da 4 bits in BCD	0	0	1	0	0	1	1	1
ANA r	A ← A ∧ r Istruzione logica AND tra il registro r e l'accumulatore.	1	0	1	0	0	S	S	S
ANI dato	A ← A ∧ dato AND di dato e accumulatore	1	1	1	0	0	1	1	0
XRA r	A ← A ⊻ r Esegue l'operazione logica di or esclusivo tra il registro r e l'accumulatore	1	0	1	0	1	S	S	S

Mnemonico	*Descrizione*	*1° byte dell'istruzione*							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
XRI dato	$A \leftarrow A \veebar$ dato Operazione or esclusivo tra dato e accumulatore	1	1	1	0	1	1	1	0
ORA r	$A \leftarrow A \vee r$ Operazione logica or tra il contenuto del registro r e l'accumulatore	1	0	1	1	0	S	S	S
ORI dato	$A \leftarrow A \vee$ dato OR logico tra il dato e l'accumulatore	1	0	1	1	1	S	S	S
CMP r	A-r sottrazione Confronta il contenuto del registro r e l'accumulatore. L'accumulatore non cambia	1	0	1	1	1	S	S	S
CPI dato	A-dato Confronta dato e accumulatore. L'accumulatore non cambia	1	1	1	1	1	1	1	0
RLC	Rotazione a sinistra dell'accumulatore $(A_n+1) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$; $(CY) \leftarrow (A_7)$	0	0	0	0	0	1	1	1
RRC	Rotazione a destra dell'accumulatore $(A_n) \leftarrow (A_n+1)$; $(A_7) \leftarrow (A_0)$; $(CY) \leftarrow (A_0)$	0	0	0	0	1	1	1	1
RAL	Rotazione dell'accumulatore a sinistra attraverso il carry $(A_n+1) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$; $(A_0) \leftarrow (CY)$	0	0	0	1	0	1	1	1
RAR	Rotazione dell'accumulatore a destra attraverso il carry $(A_n) \leftarrow (A_n+1)$; $(CY) \leftarrow (A_0)$; $(A_0) \leftarrow (CY)$	0	0	0	1	1	1	1	1
CMA	$A \leftarrow A$ Complemento dell'accumulatore	0	0	1	0	1	1	1	1
CMC	$CY \leftarrow CY$ Complemento del carry	0	0	1	1	1	1	1	1

Mnemonico	*Descrizione*	*1⁰ byte dell'istruzione*							
		D7	D6	D5	D4	D3	D2	D1	D0
STC	CY ← 1 Setta il Carry	0	0	1	1	0	1	1	1
JMP mem.	PC ← mem Il contatore di programma è caricato con l'indirizzo di memoria del 2⁰ e 3⁰ byte	1	1	0	0	0	0	1	1
JXX mem.	PC ← mem; se CCC=1 Salto condizionale; dipende dai flags	1	1	C	C	C	0	1	0
CALL mem.	SP-1 ← PCH SP-2 ← PCL SP ← SP-2 PC ← mem (3⁰ e 2⁰ byte) Salta incodizionatamente alla subroutine che occupa la locazione di memoria fornita dal 2⁰ e 3⁰ byte	1	1	0	0	1	1	0	1
CXX mem. Condizionato	Uguale a CALL se è verificata la condizione. Dipende dai flags	1	1	C	C	C	1	0	0
RET	PCL ← SP PCH ← SP+1 SP ← SP+2 Ritorno incondizionato da subroutine	1	1	0	0	1	0	0	1
RXX	Uguale a RET, se CCC=1. Dipende dai flags	1	1	C	C	C	0	0	0
RST n	SP-1 ← PCH SP-2 ← PCL SP ← SP-2 PC ← 8*NNN Salto incondizionato a una-subroutine con indirizzo n	1	1	N	N	N	1	1	1
PCHL	PC ← HL Il contenuto di HL, sostituisce il contenuto del contatore di programma	1	1	1	0	1	0	0	1

Mnemonico	*Descrizione*	*1^0 byte dell'istruzione*							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
PUSH rp	SP-1 ← rp rp ← SP+1 SP ← SP-2 Pone nello stack il contenuto della coppia di registri rp	1	1	R	P	0	1	0	1
POP rp	rl ← SP sh ← SP+1 SP ← SP+2 Trasferisce il contenuto dello stack nella coppia di registri rp	1	1	R	P	0	0	0	1
PUSH PSW	Pone nello stack il contenuto del registro P	1	1	1	1	0	1	0	1
POP PSW	Trasferisce il contenuto dello stack nel registro P	1	1	1	1	0	0	0	1
XTHL	L ↔ SP H ↔ SP+1 Il contenuto dei registri L e H viene scambiato con le prime due locazioni dello stack	1	1	1	0	0	0	1	1
SPHL	SP ← HL Il contenuto di H e L sostituisce il contenuto del puntatore di stack	1	1	1	1	1	0	0	1
IN canale	A ← (dati canale) L'ingresso *canale* invia i suoi 8 bit di informazione, che vengono depositati nell'accumulatore	1	1	0	1	1	0	1	
OUT canale	(dati canale) ← A Il contenuto dell'accumulatore viene inviato al canale di uscita	1	1	0	1	0	0	1	1

Tabella VI.-Tabella delle istruzioni del μP 8085.

Istruzioni specifiche dell'8085

Praticamente tutto quanto detto relativamente al software dell'8080 è applicabile all'8085, in quanto l'8085 è un perfezionamento dell'8080.

Esistono due istruzioni specifiche del µP 8085, cioè la RIM e la SIM con codice macchina o dell'operazione 20 e 30 esadecimale, rispettivamente. Queste istruzioni sono specifiche in quanto solo l'8085 possiede gli interrupts mascherabili, descritti precedentemente. L'istruzione RIM (Read Interrupt Mask), esegue la lettura della maschera degli interrupts, mentre la SIM (Set Interrupt Mask) scrive la maschera degli interrupts.

I \ J	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	LXI B	STAX B	INX B	INR B	DCR B	MVI B	RIC		DAD B	LDAX B	DCX B	INR C	DCR C	MVI C	RRC
1		LXI D	STAX D	INX D	INR D	DCR D	MVI D	RAI		DAD D	LDAC D	DCX D	INR E	DCR E	MVI E	RAR
2	RIM	LXI H	SHLD adr	INX H	INR H	DCR H	MVI H	DAA		DAD H	LHLD adr	DCX H	INR L	DCR L	MVI L	CMA
3	SIM	LXI SP	STA adr	INX SP	INR M	DCR M	MVI M	STC		DAD SP	LDA adr	DCX SP	INR A	DCR A	MVI A	CMC
4	MOV BB	MOV BC	MOV BD	MOV BE	MOV BH	MOV BL	MOV BM	MOV BA	MOV CB	MOV CC	MOV CD	MOV CE	MOV CH	MOV CL	MOV CM	MOV CA
5	MOV DB	MOV DC	MOV DD	MOV DF	MOV DH	MOV DL	MOV DM	MOV DA	MOV EB	MOV FC	MOV ED	MOV EE	MOV EH	MOV FL	MOV FM	MOV EA
6	MOV HB	MOV HC	MOV HD	MOV HE	MOV HH	MOV HL	MOV HM	MOV HA	MOV LB	MOV LC	MOV LD	MOV LE	MOV LH	MOV LL	MOV LM	MOV LA
7	MOV MB	MOV MC	MOV MD	MOV ME	MOV MH	MOV ML	HLT	MOV MA	MOV AB	MOV AC	MOV AD	MOV AE	MOV AH	MOV AL	MOV AM	MOV AA
8	ADD B	ADD C	ADD D	ADD E	ADD H	ADD L	ADD M	ADD A	ADC B	ADC C	ADC D	ADC E	ADC H	ADC L	ADC M	ADC A
9	SUB B	SUB C	SUB D	SUB E	SUB H	SUB L	SUB M	SUB A	SBB B	SBB C	SBB D	SBB E	SBB H	SBB L	SBB M	SBB A
A	ANA B	ANA C	ANA D	ANA E	ANA H	ANA L	ANA M	ANA A	XRA B	XRA , C	XRA D	XRA E	ARA H	XRA L	XRA M	XRA A
B	ORA B	ORA C	ORA D	ORA E	ORA H	ORA L	ORA M	ORA A	CMP B	CMP C	CMP D	CMP E	CMP H	CMP L	CMP M	CMP A
C	RNZ	POP B	JNZ adr	JMP adr	CNZ adr	PUSH B	ADI	RST 0	RZ	RET	JZ adr		CZ adr	CALL adr	ACI	RST 1
D	RNC	POP D	JNC adr	OUT adr	CNC adr	PUSH D	SUI	RST 2	RC		JC adr	JN adr	CC adr		SBI	RST 3
E	RPO	POP H	JPO adr	XTHL	CPO adr	PUSH H	ANI	RST 4	RPE		PCHI	JPE adr	XCHG	CPE adr	XRI	RST 5
F	RP	POP PSW	JP adr	DI	CP adr	PUSH PSW	ORI	RST 6	RM	SPHI	JM adr	PI	CM adr		CPI	RST 4

Tipo di istruzione	Numero dei cicli dell'istruzione	Cicli di clock	Tipo di istruzione	Numero dei cicli dell'istruzione	Cicli di clock
ALU R	1	4	LDAX	2	7
CMC	1	4	MVI	2	7
CMA	1	4	MOV M, R	2	7
DAA	1	4	MOV R, M	2	7
DCR R	1	4	STAX	2	7
DI	1	4	CALL COND.	2/5	9/18
EI	1	4	DAD	3	10
INR R	1	4	DCR	3	10
MOV R, R	1	4	IN	3	10
NOP	1	4	INR M	3	10
ROTATE	1	4	JMP	3	10
RIM	1	4	LOARD PAIR	3	10
SIM	1	4	MVI N	3	10
STC	1	4	OUT	3	10
XCHG	1	4	POP	3	10
HLT	1	5	RET	3	10
DCX	1	6	PUST	3	12
INX	1	6	RST	3	12
PCHL	1	6	LDA	2	13
RET COND.	1/3	6/12	STA	3	13
SPHL	1	6	LHLD	5	16
ALU I	2	7	SHLD	5	16
ALU M	2	7	XTHL	5	16
JNC	2/3	7/10	CALL	5	18

Tabella VIII.- Tabella delle istruzioni del μP 8085 in funzione dei tempi di esecuzione.

CAPITOLO 5

MICROPROCESSORI A 8 BIT PIU' COMUNI (II)

uesto capitolo intende fornire una panoramica delle più importanti caratteristiche del microprocessore a 8 bit della ZILOG, che è utilizzato in molti personal e home computer, tra i quali il popolare Spectrum.

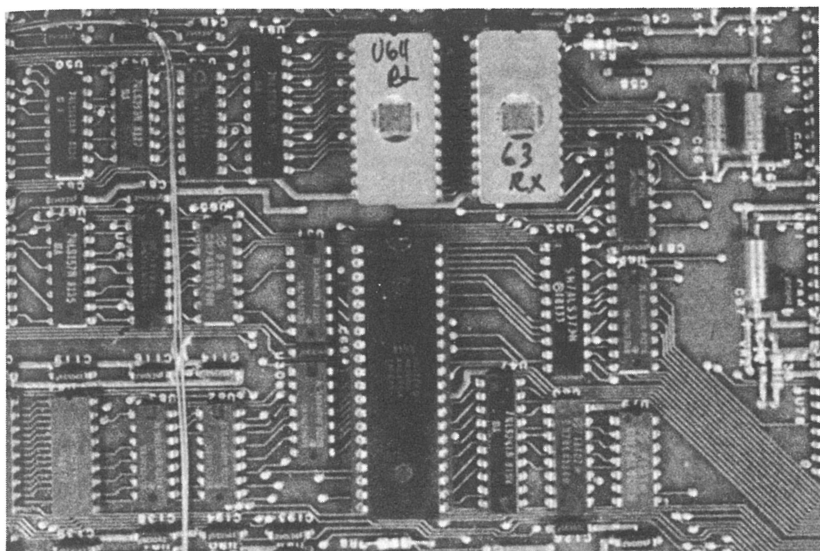



Foto 1.-Chip del microprocessore a 8 bit Z80, progettato dalla ZILOG, utilizzato come componente principale sulla piastra di un computer.

Il microprocessore Z80

E' stato posto in commercio dal 1976, e come il µP 8085, anche lo Z80 è un microprocessore simile all'8080, ma notevolmente migliorato. Anche lo Z80 è uno dei microprocessori più noti e più utilizzati, soprattutto negli home e nei personal computer, per cui verrà descritto in modo dettagliato.

Si fa rilevare che nel suo chip si trovano integrati circa ottomila transistori, quattromilacinquecento in più rispetto al predecessore 8080.

Caratteristiche dello Z80

E' stato progettato dalla ZILOG, benchè attualmente esistano diverse case che lo costruiscono come: SGS, MOSTEK, NEC e SHARP.

- è costruito in tecnologia MOS a canale N, con gate al silicio;
- è un microprocessore con 8 bit per il bus dei dati e 16 bit per quello degli indirizzi, in grado quindi di indirizzare 64 Kbytes di memoria;
- è un microprocessore veloce, che ammette un clock di 2,5 MHz; nella versione Z80-A raggiunge i 4 MHz;
- possiede 158 istruzioni, che se si considerano i diversi modi di indirizzamento, possono arrivare fino a 696;
- i metodi di indirizzamento sono: implicito, immediato, relativo, diretto, e indicizzato;
- la sua alimentazione è unica, a +5 V;
- permette due tipi di interrupts INT (Maskable Interrupt) e NMI (Not Maskable Interrupt.).

Per quanto riguarda le particolarità, il µP Z80 dispone di:

- ingressi e uscite per indirizzare fino a 256 porte di accesso alle periferiche;
- un contatore a sette bit, ed i corrispondenti circuiti logici, per ottenere le funzioni di refresh per le memorie dinamiche collegate (DRAM);
- istruzioni adatte per la manipolazione, a livello di bit, dei registri e della memoria;
- istruzioni di copia e confronto a livello di blocchi.

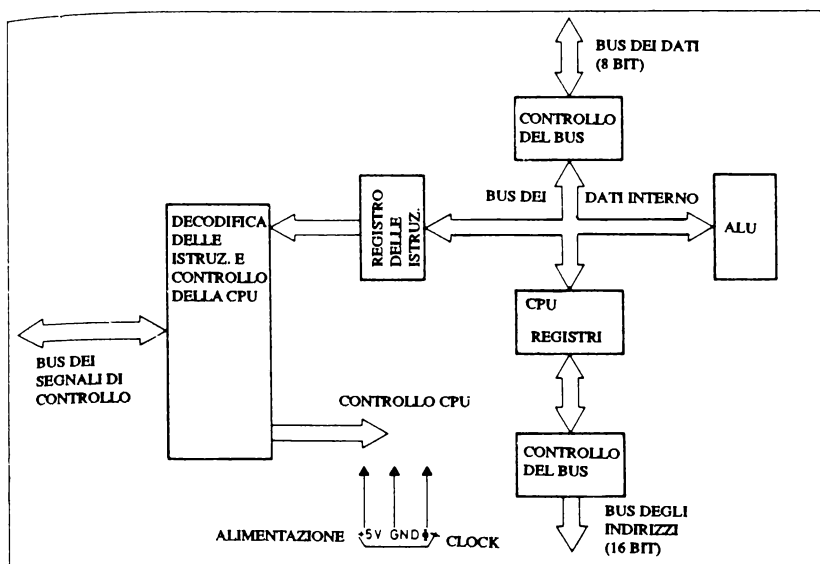



Fig. 1.-Blocchi interni del μP Z80.

Terminali dello Z80

Il μP Z80 si trova incapsulato in un contenitore a 40 piedini, come rappresentato in Fig. 2, le cui funzioni sono:

- A0-A15 costituiscono il bus degli indirizzi a 16 bit per raggiungere i 64 Kbytes di memoria;
- D0-D7 costituiscono il bus dei dati a 8 bit, attraverso il quale i dati fluiscono in entrambi i sensi tra il μP e l'esterno;
- WR (Memory Write), quando tale terminale è a zero il μP indica all'esterno l'operazione di scrittura;
- RD (Memory Read), quando tale terminale è a zero, il μP indica all'esterno l'operazione di lettura;
- MREQ (Memory Request), quando il terminale è a livello basso, indica all'esterno che l'indirizzo del bus degli indirizzi è valido per leggere o scrivere;
- M1 (Machine Cycle One), tramite questo terminale il μP segnala all'esterno il primo ciclo di esecuzione di una istruzione (ciclo di ricerca);

- HALT, se il terminale è a zero il μP indica che in tale istante si arresta, in conseguenza dell'esecuzione dell'istruzione HALT proveniente da programma;
- WAIT è l'ingresso attraverso il quale le periferiche e memorie lente arrestano il μP per sincronizzarsi con esso, fatto che avviene quando viene portato a livello zero;
- RESET, portando a zero questo ingresso, si inizializza il funzionamento del μP; ciò produce l'azzeramento del contatore di programma, in modo che quando è presente tale segnale il μP riparte dall'indirizzo 0000;
- RFSH, quando questa uscita si azzera, indica che i sette bit meno significativi del bus degli indirizzi contengono l'indirizzo di refresh per le memorie dinamiche che si possono collegare al μP;
- IORQ (Input/Output Request), se è basso indica che il μP non si rivolge alla memoria principale, ma ai circuiti di ingresso e uscita;
- INT (Interrupt Request) è un ingresso attraverso il quale le periferiche inviano una richiesta di interrupt al μP; tale richiesta avviene quando l'ingresso viene posto a zero logico;
- NMI è un ingresso di interrupt non mascherabile. Questo ingresso si attiva con il fronte di discesa del segnale di interrupt inviato a tale terminale. L'interruzione richiesta tramite NMI ha una priorità assoluta sull'interruzione INT. Quando tale interrupt (NMI) viene attivato, il μP completa l'esecuzione dell'istruzione in corso e, indipendentemente dal contenuto del registro di stato, salta alla locazione 0066 esadecimale. Questo è l'indirizzo a cui deve iniziare la subroutine di trattamento dell'interrupt;
- BUSRQ (Bus Request) è un ingresso mediante il quale si portano ad alta impedenza le uscite del bus dei dati, degli indirizzi, e le uscite di controllo, che sono tutte del tipo three state. Questa possibilità è utile per mettere in parallelo vari microprocessori che accedono indistintamente, ma non simultaneamente, agli stessi indirizzi di memoria. Il μP porta le uscite ad alta impedenza, dopo aver completato l'istruzione in corso;
- BUSAK (Bus Acknowledge) è un'uscita per indicare all'esterno che il μP è scollegato per azione dell'ingresso BUSRQ. Ciò avviene se su tale uscita è presente uno zero logico;
- Φ è l'ingresso di clock che, per questo μP, ha una sola fase;
- INGRESSI DI ALIMENTAZIONE: l'alimentazione è unica a +5 V e deve essere collegata agli ingressi 11 e 29.

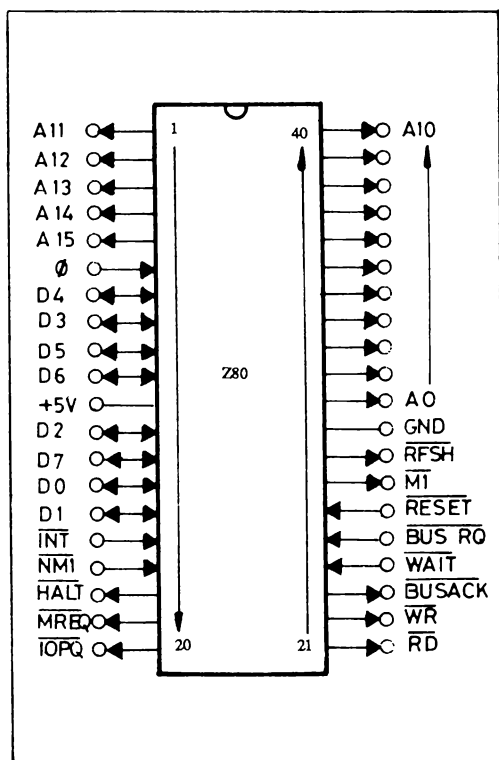



Fig. 2.-Terminali del μP Z80 e loro funzioni.

Dinamica dei segnali principali

Nei diagrammi di Fig. 3 sono rappresentate le varie combinazioni dei segnali che escono dai relativi terminali nei momenti di lettura e scrittura nella memoria, e il tempo del ciclo di ricerca dell'istruzione. Tali tempi sono forniti, in quanto rappresentativi del μP Z80, poichè durante gli stessi agisce la maggioranza dei segnali descritti. Si fa notare che nel ciclo di lettura e scrittura, i segnali attivi sono RD e WR, e nel ciclo di ricerca M1.

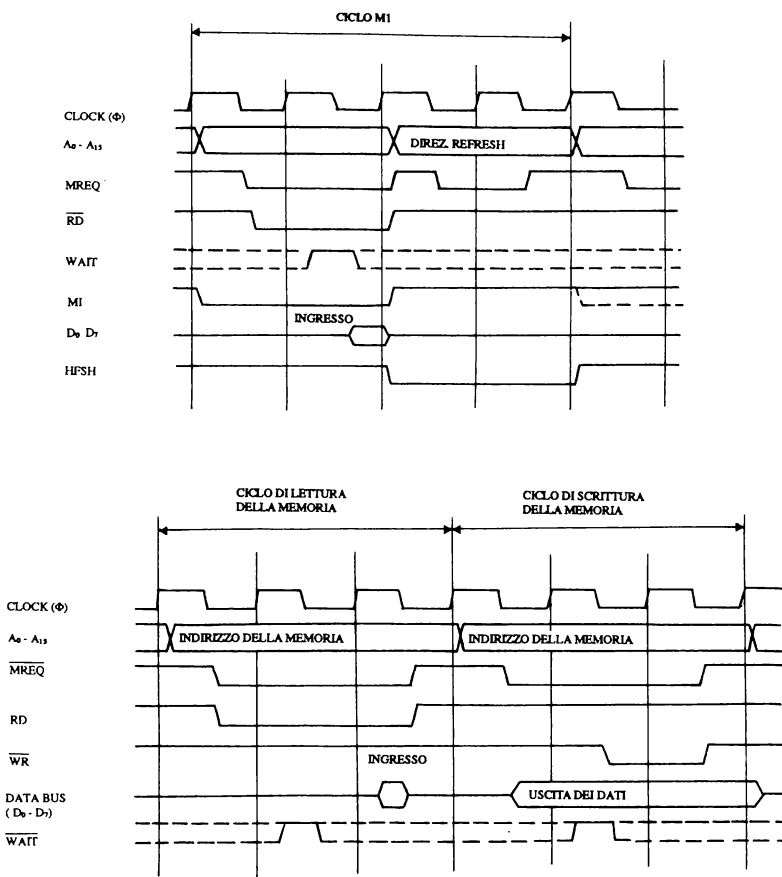



Fig. 3.-a) Diagramma dei tempi per il ciclo M1. b) Diagramma dei tempi di lettura e scrittura in memoria.

I registri interni

Questo microprocessore contiene 22 registri disponibili per l'utente. Se si osserva la Fig. 4, in cui sono rappresentati, si può notare che quelli a sinistra si trovano duplicati a destra, cioè il µP dispone di due banchi completi di registri equivalenti. Il programmatore può lavorare con l'uno o l'altro gruppo di registri, passando dall'uno all'altro mediante le opportune istruzioni di scambio. I registri di uso generale, che sono tre, possono essere utilizzati come registri a 8 o 16 bit in base al tipo di istruzione. La funzione di ogni registro è la seguente:

- registri di uso generale: sono utilizzati in base alle esigenze del programmatore, e sono i registri accumulatori A e A'.

Da programma, si indica a quale dei due si vuole accedere. I registri B, C, D, E, H e L sono registri di uso generale, e si accede ad ognuno di essi in base all'indicazione dell'istruzione utilizzata. Si dispone anche di B', C', D', E', H' e L' nell'altro banco di registri. Come detto i registri possono essere accoppiati per lavorare con 16 bit. Quando si sceglie tale soluzione i registri si accoppiano come segue: B-C, D-E, H-L e B'-C', E'-D', H'-L'.

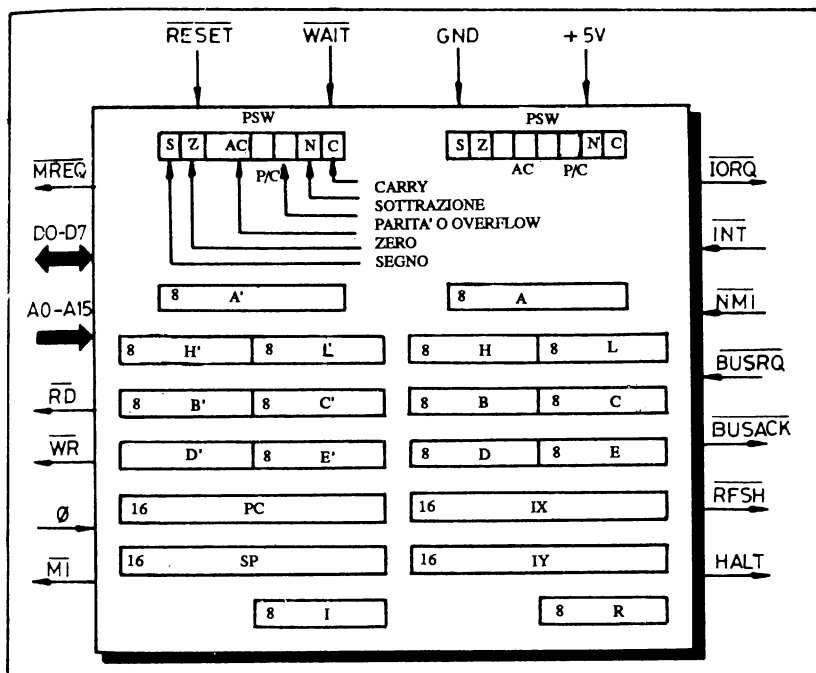



Fig. 4.-Registri interni del microprocessore Z80.

Registri di uso specifico nello Z80

Il µP Z80 dispone di quattro registri di uso specifico a 16 bit, detti PC, SP, IX, IY, ed altri due registri specifici a otto bit detti I e R. Il loro funzionamento è il seguente:

- Pc (Program Counter), è il registro contatore di programma, a 16 bit, in cui è memorizzato l'indirizzo di memoria dell'istruzione in corso. Quando ne termina l'esecuzione, il PC si incrementa automaticamente di uno. Quando il programma presenta un'istruzione di salto, ne fornisce l'indirizzo al PC, per cui il salto viene eseguito e il programma continua dal nuovo indirizzo. E' il funzionamento tipico di tutti i registri PC;
- SP (Stack Pointer), è un registro a 16 bit il cui compito è quello di funzionare da puntatore, per indicare in quale indirizzo della RAM è ubicato lo stack delle subroutines; questo stack funziona come una memoria LIFO (Last Input First Output), il che significa che l'ultimo dato memorizzato è il primo ad essere recuperato. Il µP possiede due istruzioni dette PUSH e POP che, da programma, inseriscono e recuperano dati dallo stack.;
- IX e IY sono due registri indice indipendenti tra loro, ciascuno a 16 bit. Sono utilizzati dal µP come indice, per effettuare operazioni con indirizzamento indicizzato; tale tipo di indirizzamento semplifica la stesura di un programma, specialmente quando si accede a tabelle di dati.
- il registro I, detto Interrupt Page Address Register, è un registro a otto bit e può essere utilizzato quando è necessaria una chiamata indiretta a una locazione di memoria, in seguito ad un interrupt;
- il registro R, detto Memory Refresh Register, è ad 8 bit e può essere utilizzato per accedere a memorie dinamiche. Il contenuto di tale registro è automaticamente incrementato dopo ogni istruzione di ricerca. Questi otto bit possono uscire dal µP verso l'esterno, sulla parte bassa del bus degli indirizzi, per rinfrescare continuamente la memoria RAM dinamica a cui si vuole accedere.

Il registro di stato

Il registro di stato di questo microprocessore è a 8 bit, dei quali solo sei

hanno un significato

- Bit 0=C (Carry), indica l'esistenza di un riporto nella parte più alta dell'accumulatore. E' utilizzato per le operazioni di somma e differenza, e in quelle di rotazione;
- Bit 6=Z (Zero), questo bit viene settato a uno se l'operazione eseguita ha risultato zero, oppure se è stato caricato uno zero nell'accumulatore, altrimenti rimane zero;
- Bit 7=S (Signe), tale bit indica il segno del dato contenuto nell'accumulatore, e verrà settato a uno quando il dato è negativo, a zero se il dato è positivo (un numero esadecimale è negativo se il bit 7 è uno);
- Bit 2=P/V (Parity/Overflow), questo bit può indicare due distinte funzioni: la parità di un risultato inviato all'accumulatore, quando è stata eseguita un'istruzione logica (Parità pari=1, Impari=0), e l'overflow o traboccamento quando è stata eseguita un'istruzione aritmetica;
- Bit 4=(H) è il bit che indica il carry di mezzo byte quando si opera in BCD, cioè il carry dei quattro bit meno significativi all'interno del byte;
- Bit 1=(N) indica il tipo di istruzione, somma o differenza, che è stata eseguita, operando in BCD.

Hardware di base per lo Z80

Il sistema minimo di componenti che devono essere associati al microprocessore, perchè possa funzionare in modo completo, è visibile in Fig. 5. Il μP necessita, come è noto, di un clock generato esternamente, che gli viene fornito tramite il terminale detto **Φ**.

I bus dei dati e degli indirizzi sono collegati in parallelo a tutto il sistema. Questo primo blocco deve essere formato da memorie ROM, in cui risiede il firmware specifico dell'applicazione del sistema, ed il cui indirizzo deve iniziare alla locazione 0000. Il secondo blocco è costituito dalla memoria RAM, necessaria per i diversi compiti di lettura e scrittura dei dati richiesti dal programma; nell'esempio è di soli 256 bytes. Il terzo blocco è costituito da un insieme di registri disposti in modo da costituire una porta per l'accesso alle periferiche.

La parte più interessante e specifica di questo circuito è il bus di controllo, integrato tramite i terminali detti MREQ, RD, WR, IORQ, e M1, le cui fun-

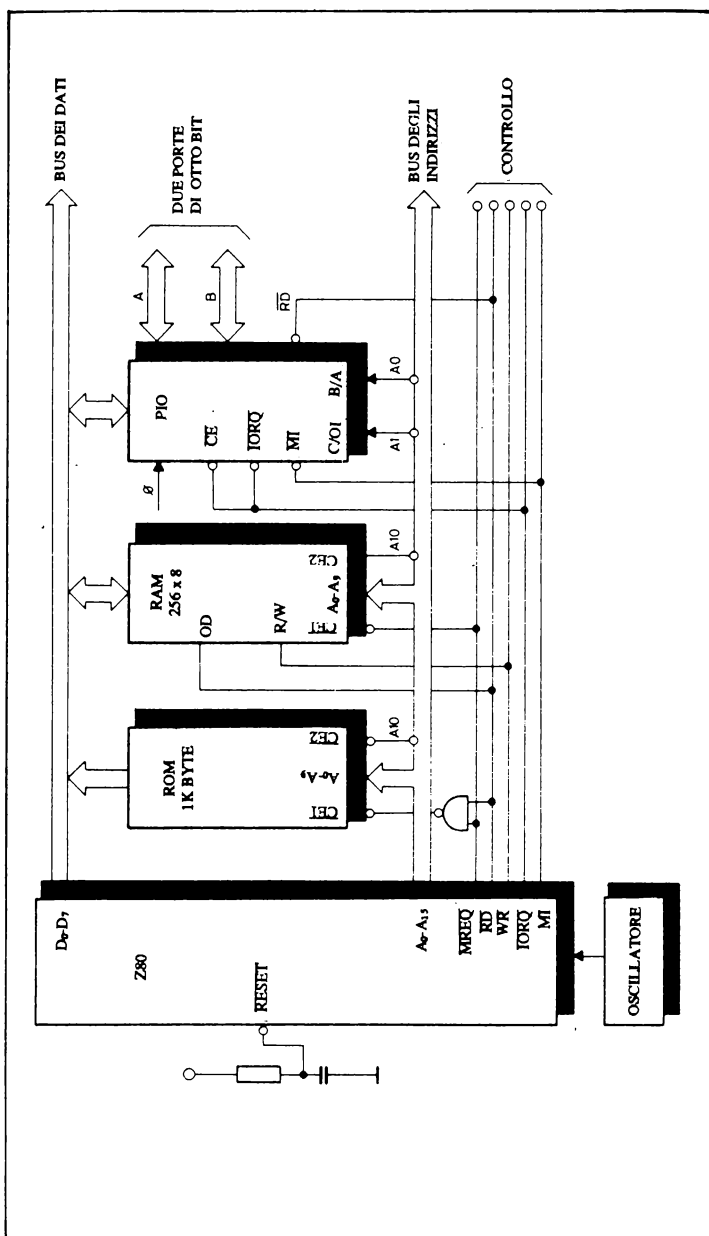



Fig. 5.-Semplice schema di collegamento del μP Z80.

zioni sono note.

MREQ è un segnale di permesso per accedere alla memoria, e andrà perciò collegato a quei blocchi che funzionano come memoria, cioè le ROM e la RAM, mentre il segnale IORQ che permette l'accesso alle periferiche, andrà collegato a quei circuiti che a queste accedono, come le PIO (Periferal Input/Output).

Questi due segnali, così disposti, distinguono un blocco dall'altro: i segnali RD e WR sono quelli che indicano se l'operazione è di scrittura o di lettura, per tanto accederanno alla memoria (alla ROM accede solo il segnale RD, in quanto può solo essere letta).

Infine il segnale M1, che è un segnale di sincronismo, dovrà essere collegato a quei circuiti per cui è necessaria una esatta sincronia con l'esecuzione dell'istruzione (si ricordi che M1 compare durante la prima parte dell'esecuzione dell'istruzione); come si vede nello schema la PIO necessita del segnale M1.

Gli interrupts nel µP Z80

Il microprocessore Z80 ha due ingressi di interrupt: l'ingresso di interrupt non mascherabile NMI e l'ingresso di interrupt mascherabile INT.

Quando viene generato l'interrupt NMI, non può essere disattivato da programma, e verrà accettato ogni volta che una periferica lo richiede. Questo interrupt sarà generalmente riservato alle funzioni più importanti, che devono necessariamente essere soddisfatte nel momento in cui si producono, come ad esempio la mancanza di alimentazione, ecc.

L'interrupt mascherabile INT può essere o meno permesso, in base alla necessità, dal programma. Questo fatto consente al programmatore di non preoccuparsi delle interruzioni nelle zone del programma che non devono essere interrotte.

Per quanto sopra, qualsiasi periferica collegata al terminale INT potrà richiedere un'interruzione in qualsiasi momento, ma questa non sarà soddisfatta se il programma ha preventivamente bloccato la possibilità di accesso.

La richiesta di interrupt potrà essere invece soddisfatta quando il programma la permetterà nuovamente.

All'interno del µP esiste un bistabile detto IFF, che può essere variato dal programma utilizzando l'istruzione EI (Enable Interrupt), per settarlo a uno, permettendo gli interrupts, e l'istruzione DI (Disable Interrupt), per resettarlo a zero, bloccando così l'ingresso di interrupt.

Uso degli interrupts

Quanto detto relativamente al bistabile IFF è valido solo concettualmente, poichè in realtà non è costituito da un solo bistabile ma da due, detti: IFF_1, che è quello che permette o meno gli interrupts, e IFF_2, che può essere considerato come una locazione di memorizzazione temporanea dello stato di IFF_1.

Quando avviene il reset generale del µP, all'accensione, questi due bistabili sono entrambi a zero e quindi gli interrupts saranno bloccati. Quando si esegue un'istruzione EI entrambi i bistabili vengono settati a uno, rendendo possibili gli interrupts. Se avviene un'interruzione, questa viene soddisfatta, ed entrambi i bistabili vengono automaticamente resettati a zero. In questo caso, ambedue funzionano all'unisono e necessitano di essere settati a uno da programma, dopo ogni trattamento di eventuali interrupts.

Lo scopo di IFF_2 è di salvaguardare il contenuto di IFF_1, prima che sia cancellato. Tale funzionamento dei bistabili permette di conoscere in ogni momento quali interrupts sono stati eseguiti, e se ne rimane qualcuno in attesa di essere soddisfatto.

Il bistabile IFF può essere reinizializzato da programma, o tramite un'istruzione di ritorno da una subroutine di interrupt non mascherabile. Si rammenta che il trattamento di un interrupt si effettua generalmente mediante un sottoprogramma.

Gli interrupts, in generale, sono utili in qualsiasi punto del programma, poichè il programmatore può dimenticarsi di tenerli presenti. Quando avviene un interrupt, l'unica cosa che dovrà essere considerata è di creare una subroutine alla fine del programma principale, nella quale questa venga adeguatamente trattata; occorrerà però bloccare gli interrupts successivi, quando se ne sta già trattando qualcuno.

Programmare tutto il sistema di un microelaboratore e il suo sistema di interrupt, è soltanto questione di pratica e di perfetta conoscenza dello stesso. Si ricorda che si stanno trattando solo concetti generali e funzioni particolari dei sistemi; per approfondire questi argomenti occorre necessariamente ricorrere ai volumi informativi forniti dai costruttori.

Tabella delle istruzioni dello Z80

Nella tabella delle istruzioni di questo microprocessore, i codici mnemonici o nomi inglesi delle istruzioni sono rappresentativi e hanno un riferimento diretto con ciascuna istruzione. I microprocessori possiedono pressochè le stesse istruzioni, ma alcune differiscono tra loro in base al criterio con cui il microprocessore è stato progettato. Nei microprocessori più evoluti si nota un

Codice dell'operazione	Mnemonico	Codice dell'operazione	Mnemonico	Codice dell'operazione	Mnemonico	Codice dell'operazione	Mnemonico
00	NOP	56	LD D, (HL)	D7	RST 10H	ED 67	RRD
01 yyy	LD BC, data 16	5 1sss	LD E, reg	D8	RET C	ED 6F	RLD
02	LD (BC), A	5E	LD E, (HL)	D9	EXX	ED A0	LDI
03	INC BC	6 0sss	LD H, reg	DA ppqq	JP C, addr	ED A1	CPI
04	INC B	66	LD H, (HL)	DB yy	IN A, (port)	ED A2	INI
05	DEC B	6 1sss	LD L, reg	DC ppqq	CALL C, addr	ED A3	OUTI
06 yy	LD B, data	6E	LD L, (HL)	DD 00xx 9	ADD IX, pp	ED A8	LDD
07	RLCA	7 0sss	LD (HL), reg	DD 21 yyyy	LD IX, data 16	ED A9	CPD
08	EX AF, AF	76	HALT	DD 22 ppqq	LD (addr), IX	ED AA	IND
09	ADD HL, DC	7 1sss	LD A, reg	DD 23	INC IX	ED AB	OUTD
0A	LD A, (BC)	7E	LD A, (HL)	DD 2A ppqq	LD IX, (addr)	ED B0	LDIR
0B	DEC BC	8 0rrr	ADD A, reg	DD 2B	DEC IX	ED B1	CPIR
0C	INC C	86	ADD A, (HL)	DD 34 disp	INC (IX+disp)	ED B2	INIR
0D	DEC C	8 1rrr	ADC A, reg	DD 35 disp	DEC (IX+disp)	ED B3	OTIR
0E yy	LD C, data	8E	ADC A, (HL)	DD 36 disp yy	LD (IX+disp), data	ED B8	LDDR
0F	RRCA	9 0rrr	SUB reg	DD 01ddd110 disp	LD reg, (IX+disp)	ED B9	CPDR
10 disp-2	DJNZ disp	96	SUB (HL)	DD 7 0ssc disp	LD (IX+disp), reg	ED BA	INDR
11 yyyy	LD DE, data 16	9 1rrr	SBC A, reg	DD 86 disp	ADD A, (IX+disp)	ED BB	OTDR
12	LD (DE), A	9 E	SBC A, (HL)	DD 8E disp	ADC A, (IX+disp)	EE yy	XOR data
13	INC DE	A 0rrr	AND reg	DD 96 disp	SUB (IX+disp)	EF	RST 28H
14	INC D	A6	AND (HL)	DD 9E disp	SBC A, (IX+disp)	F0	RET P
15	DEC D	A 1rrr	XOR reg	DD A6 disp	AND (IX+disp)	F1	POP AF
16 yy	LD D, data	AE	XOR (HL)	DD AE disp	XOR (IX+disp)	F2 ppqq	JP P, addr
17	RLA	B 0rrr	OR reg	DD B6 disp	OR (IX+disp)	F3	DI
18 disp-2	JR disp	B6	OR (HL)	DD BE disp	CP (IX+disp)	F4 ppqq	CALL P, addr
19	ADD HL, DE	B 1rrr	CP reg	DD CB disp 06	RLC (IX+disp)	F5	PUSH AF
1A	LD A, (DE)	BE	CP (HL)	DD CB disp 0E	RRC (IX+disp)	F6 yy	OR data
1B	DEC DE	C0	RET NZ	DD CB disp 16	RL (IX+disp)	F7	RST 30H
1C	INC E	C1	POP BC	DD CB disp 1E	RR (IX+disp)	F8	RET M
1D	DEC E	C2 ppqq	JP NZ, addr	DD CB disp 26	SLA (IX+disp)	F9	LD SP, HL
1E yy	LD E, data	C3 ppqq	JP addr	DD CB disp 2E	SRA (IX+disp)	FA ppqq	JP M, addr

Codice del-l'operazione	Mnemonico	Codice del-l'operazione	Mnemonico	Codice del-l'operazione	Mnemonico	Codice del-l'operazione	Mnemonico
1F	RRA	C4 ppqq	CALL NZ, addr	DD CB disp 3E	SRL (IX+disp)	FB	EI
20 disp-2	JR NZ, disp	C5	PUSH BC	DD CB dispo1bbb110	BIT b, (IX+disp)	FC ppqq	CALL M, addr
21 yyyy	LD HL, data 16	C6 yy	ADD A, data	DD CB disp10bbb110	RES b, (IX+disp)	FD 00xx9	ADD 1Y, rr
22 ppqq	LD (addr), HL	C7	RST 00H	DD CB disp11bbb110	SET b, (IX+disp)	FD 21 yyyy	LD 1Y, data 16
23	INC HL	C8	RET Z	DD E1	POP IX	FD 22 ppqq	LD (addr), 1Y
24	INC H	C9	RET	DD E3	EX (SP), IX	FD 23	INC 1Y
25	DEC H	CA ppqq	JP z, addr	DD E5	PUSH IX	FD 2A ppqq	LD 1Y, (addr)
26 yy	LD H, data	CB 0 0rrr	RLC reg	DD E9	JP (IX)	FD 2B	DEC 1Y
27	DAA	CB 06	RLC (HL)	DD F9	LD SP, IX	FD 34 disp	INC (1Y+disp)
28 disp-2	JR Z, disp	CB 0 1rrr	RRC reg	DD yy	SBC A, data	FD 35 disp	DEC (1+disp)
29	ADD HL, HL	CB 0E	RRC (HL)	DF	RST 18H	FD 36 disp yy	LD (1Y+disp), data
2A ppqq	LD HL, (addr)	CB 1 0rrr	RL reg	E0	RET PO	FD 01ddd110 disp	LD reg, (1Y+disp)
2B	DEC HL	CB 16	RL (HL)	E1	POP HL	FD 7 0sss disp	LD (1Y+disp) reg
2C	INC L	CB 1 1rrr	RR reg	E2 ppqq	JP PO, addr	FD 86 disp	ADD A, (1Y+disp)
2D	DEC L	CB 1E	RR (HL)	E3	EX (SP), HL	FD 8E disp	ADC A, (1Y+disp)
2E	LD L, data	CB 1 0rrr	SLA reg	E4 ppqq	CALL PO, addr	FD 96 disp	SUB (1Y+disp)
2F	CPL	CB 26	SLA (HL)	E5	PUSH HL	FD 9E disp	SBC A, (1Y+disp)
30 disp-2	JR NC, disp	CB 2 1rrr	SRA reg	E6 yy	AND data	FD A6 disp	AND (1Y+disp)
31 yyyy	LD SP, data 16	CB 2E	SRA (HL)	E7	RST 20H	FD AE disp	XOR (1Y+disp)
32 ppqq	LD (addr), A	CB 3 1rrr	SRL reg	E8	RET PE	FD B6 disp	OR (1Y+disp)
33	INC SP	CB 3E	SRL (HL)	E9	JP (HL)	FD BE disp	CP (1Y+disp)
34	INC (HL)	CB 01bbbrrr	BIT b, reg	EA ppqq	JP PE, addr	FD CB disp 06	RLC (1Y+disp)
35	DEC (HL)	CB 01bbb110	BIT b, (HL)	EB	EX DE, HL	FD CB disp 0E	RRC (1Y+disp)
36 yy	LD (HL), data	CB 10bbbrrr	RES b, reg	EC ppqq	CALL PE, addr	FD CB disp 16	RL (1Y+disp)
37	SCF	CB 10bbb110	RES b, (Hl)	ED 01ddd000	IN reg, (C)	FD CB disp 1E	RR (1Y+disp)
38	JR C, disp	CB 11bbbrrr	SET b, reg	ED 01sss001	OUT (C), reg	FD CB disp 26	SLA (1Y+disp)
39	ADD HL, SP	CB 11bbb110	SET b, (HL)	ED 01xx2	SBC HL, rp	FD CB disp 2E	SRA (1Y+disp)
3A ppqq	LD A, (addr)	CC ppqq	CALL Z, addr	ED 01xx3 ppqq	LD (addr), rp	FD CB disp 3E	SRL (1Y+disp)
3B	DEC SP	CD ppqq	CALL addr	ED 44	NEG	FD CB disp01bbb110	BIT b, (1Y+disp)
3C	INC A	CE yy	ADC A, data	ED 45	RETN	FD CB disp10bbb110	RES b, (1Y+disp)
3D	DEC A	CF	RST 08H	ED 010nn110	IM m	*FD CB disp11bbb110*	*SET b, (1Y+disp)*

Codice dell'operazione	Mnemonico		Codice dell'operazione	Mnemonico		Codice dell'operazione	Mnemonico		Codice dell'operazione	Mnemonico	
3E yy	LD	A, data	D0	RET	NC	ED 47	LD	I, A	FD E1	POP	IY
3F	CCF		D1	POP	DE	ED 01xx A	ADC	HL, rp	FD E3	EX	(SP), IY
4 0sss	LD	B, reg	D2 ppqq	JP	NC, addr	ED 01xx B ppqq	LD	rp, (addr)	FD E5	PUSH	IY
46	LD	B, (HL)	D3 yy	OUT	(port), A	ED 4D	RETI		FD E9	JP	(IY)
4 1sss	LD	C, reg	D4 ppqq	CALL	NC, addr	ED 4F	LD	R, A	FD F9	LD	SP, IY
4E	LD	C, (HL)	D5	PUSH	DE	ED 57	LD	A, I	FE yy	CP	data
5 0sss	LD	D, reg	D6 yy	SUB	data	ED ED 5F	LD	A, R	FF	RST	38H

Il µP Z-80 possiede un elevato numero di istruzioni. Di seguito è riportata la tabella che le contiene, suddivise nei vari gruppi: di caricamento, di scambio e trasferimento, aritmetiche e logiche, di controllo della CPU, di rotazione, e infine, di selezione, cancellazione e test.

Gruppo delle istruzioni di caricamento (LOAD) di bits

Mnemonico	Simbolo della operazione	Registro di stato								Codice macchina				N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
		7	6	5	4	3	2	1	0								
		S	Z		H		P/V	N	C	76	543	210	HEX				
LD r, s	r←s	●	●	X	●	X	●	●	●	01	r	s		1	1	4	r, s Reg.
LD r, n	r←n	●	●	X	●	X	●	●	●	00	r	110		2	2	7	000 B
										←	n	→					001 C
LD r, (HL)	r←(HL)	●	●	X	●	X	●	●	●	01	r	110		1	2	7	010 D
LD r, (IX+d)	r←(IX+d)	●	●	X	●	X	●	●	●	11	011	101	DD	3	5	19	011 E
										01	r	110					100 H
										←	d	→					101 L

Mnemonico	Simbolo della operazione	Registro di stato								Codice macchina				N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
		7	6	5	4	3	2	1	0								
		S	Z		H		P/V	N	C	76	543	210	HEX				
LD r, (IY+d)	r←(IY+d)	●	●	X	●	X	●	●	●	11	111	101	FD	3	5	19	111 A
										01	r	110					
										←	d	→					
LD (HL), r	(HL)←r	●	●	X	●	X	●	●	●	01	110	r		1	2	7	
LD (IX+d), r	(IX+d)←r	●	●	X	●	X	●	●	●	11	011	101	DD	3	5	19	
										01	110	r					
										←	d	→					
LD (IY+d), r	(IY+d)←r	●	●	X	●	X	●	●	●	11	111	101	FD	3	5	19	
										01	110	r					
										←	d	→					
LD (HL), n	(HL)←n	●	●	X	●	X	●	●	●	00	110	110	36	2	3	10	
										←	n	→					
LD (IX+d), n	(IX+d)←n	●	●	X	●	X	●	●	●	11	011	101	DD	4	5	19	
										00	110	110	36				
										←	d	→					
										←	n	→					
LD (IY+d), n	(IY+d)←n	●	●	X	●	X	●	●	●	11	111	101	FD	4	5	19	
										00	110	110	36				
										←	d	→					
										←	n	→					
LD A, (BC)	A←(BC)	●	●	X	●	X	●	●	●	00	001	010	0A	1	2	7	
LD A, (DE)	A←(DE)	●	●	X	●	X	●	●	●	00	011	010	1A	1	2	7	
LD A, (nn)	A←(nn)	●	●	X	●	X	●	●	●	00	111	010	3A	3	4	13	
										←	n	→					
										←	n	→					
LD (BC), A	(BC)←A	●	●	X	●	X	●	●	●	00	000	010	02	1	2	7	
LD (DE), A	(DE)←A	●	●	X	●	X	●	●	●	00	010	010	12	1	2	7	
LD (nn), A	(nn)←A	●	●	X	●	X	●	●	●	00	110	010	32	3	4	13	
										←	n	→					
										←	n	→					

Mnemonico	Simbolo della operazione	Registro di stato 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Codice macchina 76	543	210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
LD A, I	A←I	↕	↕	X	0	X	IFF	0	●	11	101	101	ED	2	2	9	
										01	010	111	57				
LD A, R	A←R	↕	↕	X	0	X	IFF	0	●	11	101	101	ED	2	2	9	
										01	011	111	5F				
LD I, A	I←A	●	●	X	●	X	●	●	●	11	101	101	ED	2	2	9	
										01	000	111	47				
LD R, A	R←A	●	●	X	●	X	●	●	●	11	101	101	ED	2	2	9	
										01	001	111	4F				
LD dd, nn	dd←nn	●	●	X	●	X	●	●	●	00	dd0	001		3	3	10	dd PARI
										←	n	→					00 BC
										←	n	→					01 DE
LD IX, nn	IX←nn	●	●	X	●	X	●	●	●	11	011	101	DD	4	4	14	10 HL
										00	100	001	21				11 SP
										←	n	→					
										←	n	→					
LD IY, nn	IY←nn	●	●	X	●	X	●	●	●	11	111	101	FD	4	4	14	
										00	100	001	21				
										←	n	→					
										←	n	→					
LD HL, (nn)	H←(nn+1) L←(nn)	●	●	X	●	X	●	●	●	00	101	010	2A	3	5	16	
										←	n	→					
										←	n	→					
LD dd, (nn)	dd_H←(nn+1) dd_L←(nn)	●	●	X	●	X	●	●	●	11	101	101	ED	4	6	20	
										01	dd1	011					
										←	n	→					
										←	n	→					

Mnemonico	Simbolo della operazione	Registro di stato								Codice macchina				N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
		7	6	5	4	3	2	1	0								
		S	Z		H		P/V	N	C	76	543	210	HEX				
LD IX, (nn)	IX_H ← (nn+1)	●	●	X	●	X	●	●	●	11	011	101	DD	4	6	20	
	IX_L ← (nn)									00	101	010	2A				
										←	n	→					
										←	n	→					
LD IY, (nn)	IY_H ← (nn+1)	●	●	X	●	X	●	●	●	11	111	101	FD	4	6	20	
	IY_L ← (nn)									00	101	010	2A				
										←	n	→					
										←	n	→					
LD (nn), HL	(nn+1) ← H	●	●	X	●	X	●	●	●	00	100	010	22	3	5	16	
	(nn) ← L									←	n	→					
										←	n	→					
LD (nn), dd	(nn+1) ← dd_H	●	●	X	●	X	●	●	●	11	101	101	ED	4	6	20	
	(nn) ← dd_L									01	dd0	011					
										←	n	→					
										←	n	→					
LD (nn), IX	(nn+1) ← IX_H	●	●	X	●	X	●	●	●	11	011	101	DD	4	6	20	
	(nn) ← IX_L									00	100	010	22				
										←	n	→					
										←	n	→					
LD (nn), IY	(nn+1) ← IY_H	●	●	X	●	X	●	●	●	11	111	101	FD	4	6	20	
	(nn) ← IY_L									00	100	010	22				
										←	n	→					
										←	n	→					
LD SP, HL	SP ← HL	●	●	X	●	X	●	●	●	11	111	001	F9	1	1	6	
LD SP, IX	SP ← IX	●	●	X	●	X	●	●	●	11	011	101	DD	2	2	10	
										11	111	001	F9				
LD SP, IY	SP ← IY	●	●	X	●	X	●	●	●	11	111	101	FD	2	2	10	
										11	111	001	F9				qq PARI

Mnemonico	Simbolo della operazione	Registro di stato								Codice macchina				N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
		7	6	5	4	3	2	1	0								
		S	Z		H		P/V	N	C	76	543	210	HEX				
PUSH qq	$(SP\text{-}2) \leftarrow qq_L$	•	•	X	•	X	•	•	•	11	qq0	101		1	3	11	00 BC
	$(SP\text{-}1) \leftarrow qq_H$																01 DE
PUSH IX	$(SP\text{-}2) \leftarrow IX_L$	•	•	X	•	X	•	•	•	11	011	101	DD	2	4	15	10 HL
	$(SP\text{-}1) \leftarrow IX_H$									11	100	101	E5				11 AF
PUSH IY	$(SP\text{-}2) \leftarrow IY_L$	•	•	X	•	X	•	•	•	11	111	101	FD	2	4	15	
	$(SP\text{-}1) \leftarrow IY_H$									11	100	101	E5				
POP qq	$qq_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11	qq0	001		1	3	10	
	$qq_L \leftarrow (SP)$																
POP IX	$IX_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11	011	101	DD	2	4	14	
	$IX_L \leftarrow (SP)$									11	100	001	E1				
POP IY	$IY_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11	111	101	FD	2	4	14	
	$IY_L \leftarrow (SP)$									11	100	001	E1				

r,s = Rappresentano qualsiasi registro A, B, C, D, E, H, L.
• = Bit di stato non influenzato, 0 = Bit azzerato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Bit influenzato secondo il risultato dell'operazione.
dd = Rappresenta qualsiasi coppia di resgistri BC, DE, HL, SP.
qq = Rappresenta le coppie di registri AF, BC, DE, HL.
BCL, ad esempio, rappresenta un registro a 8 bit della coppia BC, in questo caso, il C, poichè L indica l'ottetto meno significativo.

Gruppo di istruzioni di scambio, trasferimento e scelta

Mnemonico	Simbolo della operazione	Registro di stato 7	6	5	4	3	2	1	0	Codice macchina 76	543	210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
		S	Z		H		P/V	N	C	76	543	210	HEX				
EX DE, HL	DE↔HL	●	●	X	●	X	●	●	●	11	101	011	EB	1	1	4	Il banco dei registri principali e ausiliari si scambiano.
EX AF, AF'	AF↔AF'	●	●	X	●	X	●	●	●	00	001	000	08	1	1	4	
EXX	(BC↔BC' DE↔DE' HL↔HL')	●	●	X	●	X	●	●	●	11	011	001	D9	1	1	4	
EX (SP), HL	H↔(SP+1) L↔(SP)	●	●	X	●	X	●	●	●	11	100	011	E3	1	5	19	
EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	●	●	X	●	X	●	●	●	11 11	011 100	101 011	DD E3	2	6	23	
EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	●	●	X	●	X	●	●	●	11 11	111 100	101 011	FD E3	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC BC·1	●	●	X	0	X	↕ [1]	0	●	11 10	101 100	101 000	ED A0	2	4	16	Carica (HL) in (DE), incrementa i puntatori, decrementa (BC).
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC·1 Repeat until BC=0	●	●	X	0	X	0	0	●	11 10	101 110	101 000	ED B0	2 2	5 4	21 16	Se BC≠0 Se BC=0
LDD	(DE)←(HL) DE←DE·1 HL←HL·1 BC←BC·1	●	●	X	0	X	↕ [1]	0	●	11 10	101 101	101 000	ED A8	2	4	16	
LDDR	(DE)←(HL) DE←DE·1 HL←HL·1	●	●	X	0	X	0	0	●	11 10	101 111	101 000	ED B8	2 2	5 4	21 16	Se BC≠0 Se BC=0

Gruppo di istruzioni di scambio, trasferimento e scelta

Mnemonico	*Simbolo della operazione*	7	6	5	4	3	2	1	0	*Codice macchina* 76	543	210	HEX	*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
		S	Z		H		P/V	N	C								
	BC ← BC-1 Repeat until BC=0																
CPI	A-(HL) HL ← HL+1 BC ← BC-1	↕	↕ [2]	X	↕	X	↕ [1]	1	•	11 10	101 100	101 001	ED A1	2	4	16	
CPIR	A-(HL) HL ← HL+1 BC ← BC-1 Repeat until A=(HL) or BC=0	↕	↕ [2]	X	↕	X	↕ [1]	1	•	11 10	101 110	101 001	ED B1	2 2	5 4	21 16	Se BC≠0 e A≠(HL) Se BC=0 e A=(HL)
CPD	A-(HL) HL ← HL-1 BC ← BC-1	↕	↕ [2]	X	↕	X	↕ [1]	1	•	11 10	101 101	101 001	ED A9	2	4	16	
CPDR	A-(HL) HL ← HL-1 BC ← BC-1 Repeat until A=(HL) or BC=0	↕	↕ [2]	X	↕	X	↕ [1]	1	•	11 10	101 111	101 001	ED B9	2 2	5 4	21 16	Se BC≠0 e A≠(HL) Se BC=0 e A=(HL)

Note:
- Il bit P/V è posto a zero se il risultato di BC-1=0, altrimenti P/V è 1. Il bit Z è 1 se A=(HL), altrimenti Z=0.
• = Bit di stato non influenzato, 0 = Bit azzerato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Bit influenzato secondo il risultato dell'operazione.

Mnemonico	Simbolo della operazione	Registro di stato 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Codice macchina 76 543 210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 000 r		1	1	4	r Reg.
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 000 110		2	2	7	000 B
										← n ←					001 C
															010 D
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 000 110		1	2	7	011 E
ADD A, (IX+d)	A ← A + (IX+d)	↕	↕	X	↕	X	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 000 110					101 L
										← d ←					111 A
ADD A, (IY+d)	A ← A + (IY+d)	↕	↕	X	↕	X	V	0	↕	11 111 101	FD	3	5	19	
										10 000 110					
										← d ←					
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	001					
SUB s	A ← A·s	↕	↕	X	↕	X	V	1	↕	010					
SBC A, s	A ← A·s·CY	↕	↕	X	↕	X	V	1	↕	011					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	100					
OR s	A ← A V s	↕	↕	X	0	X	P	0	0	110					
XOR s	A ← A (+)s	↕	↕	X	0	X	P	0	0	101					
CP s	A·s	↕	↕	X	↕	X	V	1	↕	111					
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r 100		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 100		1	3	11	
INC (IX+d)	(IX+d) ← (IX+d) + 1	↕	↕	X	↕	X	V	0	•	11 011 101	DD	3	6	23	
										00 110 100					
										→ d →					
INC (IY+d)	(IY+d) ← (IY+d) + 1	↕	↕	X	↕	X	V	0	•	11 111 101	FD	3	6	23	
										00 110 100					
										→ d →					
DEC s	s ← s·1	↕	↕	X	↕	X	V	1	•	101					

Gruppo di istruzioni di controllo della CPU

Mnemonico	*Simbolo della operazione*	*Registro di stato* 7	6	5	4	3	2	1	0	*Codice macchina* 76	543	210	HEX	*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
		S	Z		H		P/V	N	C								
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	●	↕	00	100	111	27	1	1	4	Regolazione decimale dell'accumulatore.
CPL	$A \leftarrow \bar{A}$	●	●	X	1	X	●	1	●	00	101	111	2F	1	1	4	
NEG	$A \leftarrow \bar{A}+1$	↕	↕	X	↕	X	V	1	↕	11	101	101	ED	2	2	8	
										01	000	100	44				
CCF	$CY \leftarrow \overline{CY}$	●	●	X	X	X	●	0	↕	00	111	111	3F	1	1	4	Complementa
SCF	$CY \leftarrow 1$	●	●	X	0	X	●	0	1	00	110	111	37	1	1	4	l'accumulatore.
NOP	No operation	●	●	X	●	X	●	●	●	00	000	000	00	1	1	4	Pone a 1 il Bit carry.
HALT	CPU halted	●	●	X	●	X	●	●	●	01	110	110	76	1	1	4	
DI*	$IFF \leftarrow 0$	●	●	X	●	X	●	●	●	11	110	011	F3	1	1	4	
EI*	$IFF \leftarrow 1$	●	●	X	●	X	●	●	●	11	111	011	FB	1	1	4	
IM 0	Set interrupt mode 0	●	●	X	●	X	●	●	●	11	101	101	ED	2	2	8	
										01	000	110	46				
IM 1	Sep interrupt mode 1	●	●	X	●	X	●	●	●	11	101	101	ED	2	2	8	
										01	010	110	56				
IM 2	Sep interrupt mode 2	●	●	X	●	X	●	●	●	11	101	101	ED	2	2	8	
										01	011	110	5E				

Note:
Il simbolo V nel bit P/V indica che tale bit contiene l'overflow del risultato dell'operazione. Analogamente il simbolo P indica la parità. V = 1 significa overflow, V = 0 significa non overflow. P = 1 significa parità pari. P = 0 significa disparità.
IFF indica il permesso di interrupt.
CY indica il bit di carry.
● = Bit di stato non influenzato, 0 = Bit azzerato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Bit influenzato secondo il risultato dell'operazione.

Gruppo di istruzioni aritmetiche a 16 bit

Mnemonico	*Simbolo della operazione*	*Registro di stato*								*Codice macchina*				*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
		7	6	5	4	3	2	1	0								
		S	Z		H		P/V	N	C	76	543	210	HEX				
ADD HL, ss	HL ← HL+ss	●	●	X	X	X	●	0	↕	00	ss1	001		1	3	11	ss Reg.
																	00 BC
ADC HL, ss	HL←HL+ss+CY	↕	↕	X	X	X	V	0	↕	11	101	101	ED	2	4	15	01 DE
										01	ss1	010					10 HL
																	11 SP
SBC HL, ss	HL←HL·ss·CY	↕	↕	X	X	X	V	1	↕	11	101	101	ED	2	4	15	
										01	ss0	010					
ADD IX, pp	IX←IX+pp	●	●	X	X	X	●	0	↕	11	011	101	DD	2	4	15	pp Reg.
										00	pp1	001					00 BC
																	01 DE
																	10 IX
																	11 SP
ADD IY, rr	IY←IY+rr	●	●	X	X	X	●	0	↕	11	111	101	FD	2	4	15	rr Reg
										00	rr1	001					00 BC
																	01 DE
																	10 IY
																	11 SP
INC ss	ss←ss+1	●	●	X	●	X	●	●	●	00	ss0	011		1	1	6	
INC IX	IX←IX+1	●	●	X	●	X	●	●	●	11	011	101	DD	2	2	10	
										00	100	011	23				
INC IY	IY←IY+1	●	●	X	●	X	●	●	●	11	111	101	FD	2	2	10	
										00	100	011	23				
DEC ss	ss←ss·1	●	●	X	●	X	●	●	●	00	ss1	011		1	1	6	
DEC IX	IX←IX·1	●	●	X	●	X	●	●	●	11	011	101	DD	2	2	10	
										00	101	011	2B				
DEC IY	IY←IY·1	●	●	X	●	X	●	●	●	11	111	101	FD	2	2	10	
										00	101	011	2B				

Note:
- ss: Indica qualsiasi coppia di registri BC, DE, HL, SP. - pp: indica qualsiasi coppia di registri BC, DE, IX, SP. - rr: Indica qualsiasi coppia di registri BC, DE, IX, SP. - ● = Bit di stato non influenzato, 0 = Bit azzerato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Bit influenzato secondo il risultato dell'operazione.

Gruppo di istruzioni di trattamento delle subroutines

Mnemonico	Simbolo della operazione	7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Codice macchina 76	543	210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
CALL nn	(SP-1) ← PCH	•	•	X	•	X	•	•	•	11	001	101	CD	3	5	17	
	(SP-2) ← PCL									←	n	←					
	PC ← nn									←	n	←					
CALL cc, nn	Se cc è falso continua, se no esegue CALL nn	•	•	X	•	X	•	•	•	11	cc	100		3	3	10	Se cc è falso.
										←	n	←					
										←	n	←		3	5	17	Se cc è vero
RET	PCL ← (SP)	•	•	X	•	X	•	•	•	11	001	001	C9	1	3	10	
	PCH ← (SP+1)																
RET cc	Se cc è falso continua, se no esegue RET	•	•	X	•	X	•	•	•	11	cc	000		1	1	5	Se cc è falso.
														1	3	11	Se cc è vero
RETI	Ritorno da interrupt	•	•	X	•	X	•	•	•	11	101	101	ED	2	4	14	
										01	00	101	4D				
RETN[1]	Ritorno da interrupt non mascherato	•	•	X	•	X	•	•	•	11	101	101	ED	2	4	14	
										01	000	101	45				
RST p	(SP-1) ← PCH	•	•	X	•	X	•	•	•	11	t	111		1	3	11	
	(SP-2) ← PCL																
	PCH ← 0																
	PCL ← p																

cc	Condizione
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO imparità
101	PE parità pari
110	P segno pos.
111	M segno neg.

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Note:
- [1]RETN carica IFF_2 in IFF_1. - • = Bit di stato non influenzato, 0 = Bit azzerato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Bit influenzato secondo il risultato dell'operazione.

Gruppo di istruzioni di salto

Mnemonico	*Simbolo della operazione*	*Registro di stato* 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	*Codice macchina* 76 543 210	HEX	*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
JP nn	PC←nn	●	●	X	●	X	●	●	●	11 000 011 ← n ← ← n ←	C3	3	3	10	
JP cc, nn	Si cc es **Se cc è vero** PC←nn, **se no continua**	●	●	X	●	X	●	●	●	11 cc 010 ← n ← ← n ←		3	3	10	cc — Condizione 000 NZ non zero 001 Z zero 010 NC non carry 011 C carry 100 PO imparità 101 PE parità pari 110 P segno pos. 111 M segno neg.
JR e	PC←PC+e	●	●	X	●	X	●	●	●	00 011 000 ← e-2 ←	18	2	3	12	
JR C, e	**Se C=0, continua**	●	●	X	●	X	●	●	●	00 111 000 ← e-2 ←	38	2	2	7	**Se la condizione è vera.**
	Se C=1 PC←PC+e											2	3	12	**Se la condizione è vera.**
JR NC, e	**Se C=1, continua**	●	●	X	●	X	●	●	●	00 110 000 ← e-2 ←	30	2	2	7	**Se la condizione è vera.**
	Se C=0, PC←PC+e											2	3	12	**Se la condizione è vera.**
JR Z, e	**Se Z=0, continua**	●	●	X	●	X	●	●	●	00 101 000 ← e-2 ←	28	2	2	7	**Se la condizione è vera.**
	Se Z=1, PC←PC+e											2	3	12	**Se la condizione è vera.**
JR NZ, e	**Se Z=1, continua**	●	●	X	●	X	●	●	●	00 100 000 ← e-2 ←	20	2	2	7	**Se la condizione è vera.**

Gruppo di istruzioni di salto

Mnemonico	*Simbolo della operazione*	*Registro di stato*								*Codice macchina*				*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
		7	*6*	*5*	*4*	*3*	*2*	*1*	*0*								
		S	*Z*		*H*		*P/V*	*N*	*C*	*76*	*543*	*210*	*HEX.*				
	Se Z=0, PC←PC+e													2	3	12	**Se la condizione è vera.**
JP (HL)	PC←HL	●	●	X	●	X	●	●	●	11	101	001	E9	1	1	4	
JP (IX)	PC←IX	●	●	X	●	X	●	●	●	11	011	101	DD	2	2	8	
										11	101	001	E9				
JP (IY)	PC←IY	●	●	X	●	X	●	●	●	11	111	101	FD	2	2	8	
										11	101	001	E9				
DJNZ, e	B←B·1	●	●	X	●	X	●	●	●	00	010	000	10	2	2	8	**Se BC≠0**
	Se B=0, continua									← e·2 ←							
	Se B≠0, PC←PC+e													2	3	13	**Se BC=0**

Note:
- e: Rappresenta l'estensione nell'indirizzamento relativo, è un numero in complemento a due compreso tra -126 e 129.
- e-2: In codice macchina produce un indirizzamento del PC, incrementando di due il PC prima di sommare e.

Gruppo di istruzioni di ingresso/uscita

Mnemonico	Simbolo della operazione	7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Codice macchina 76	543	210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
IN A, (n)	A←(n)	•	•	X	•	X	•	•	•	11 ←	011 n	011 ←	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r←(C) **Se r=110, solo C sarà influenzato**	↕	↕	X	↕	X	P	0	•	11 01	101 r	101 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL)←(C) B←B·1 HL←HL+1	X	① ↕	X	X	X	X	1	•	11 10	101 100	101 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL)←(C) B←B·1 HL←HL+1 **ripetere fino a che** B=0	X	1	X	X	X	X	1	•	11 10	101 110	101 010	ED B2	2 2	5 (IF B≠0) 4	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL)←(C) B←B·1 HL←HL·1	X	① ↕	X	X	X	X	1	•	11 10	101 101	101 010	ED AA	2	4	16	C to $A_0 \sim AA_7$ B to $A_8 \sim A_{15}$
INDR	(HL)←(C) B←B·1 HL←H·1 **ripetere fino a che** B=0	X	1	X	X	X	X	1	•	11 10	101 111	101 010	ED BA	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n)←A	•	•	X	•	X	•	•	•	11	010	011	D3	2	3	11	*n to $A_0 \sim A_7$* *Acc to $A_8 \sim A_{15}$*

Gruppo di istruzioni di ingresso/uscita

Mnemonico	Simbolo della operazione	Registro di stato 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Codice macchina 76	543	210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
OUT (C), r	(C)←r	●	●	X	●	X	●	●	●	11 01	101 r	101 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C)←(HL) B←B·1 HL←HL+1	X	① ↕	X	X	X	X	1	●	11 10	101 100	101 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C)←(HL) B←B·1 HL←HL+1 **ripetere fino a che** B=0	X	1	X	X	X	X	1	●	11 10	101 110	101 011	ED B3	2 2	5 (If B≠0) 4	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C)←(HL) B←B·1 HL←HL·1	X	① ↕	X	X	X	X	1	●	11 10	101 101	101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C)←(HL) B←B·1 HL←HL· **ripetere fino a che** B=0	X	1	X	X	X	X	1	●	11 10	101 111	101 011	ED BB	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Note:
1 Significa che il bit Z è settato a 1 se il risultato di B-1 è zero, altrimenti verrà resettato a 0.
● = Significa bit di stato non influenzato, 0 = Bit resettato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Il bit è influenzato secondo il risultato dell'operazione.

Gruppo di istruzioni di spostamento e rotazione

Mnemonico	*Simbolo della operazione*	*Registro di stato* 7 *S*	6 *Z*	5	4 *H*	3	2 *P/V*	1 *N*	0 *C*	*Codice macchina* 76 543 210	*HEX*	*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
RLCA	CY ← 7←0 ↵ A	•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	**Rotazione circolare a sinistra dell'accumulatore.**
RLA	↳CY ← 7←0 ↵ A	•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	**Rotazione a sinistra dell'accumulatore.**
RRCA	↳7→0 → CY A	•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	**Rotazione circolare a destra dell'accumulatore.**
RRA	↳7→0 → CY↵ A	•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	**Rotazione a destra dell'accumulatore.**
RLCr	A CY ← 7←0 ↵ r,(HL),(IX+d),(IY+d)	↕	↕	X	0	X	P	0	↕	11 001 011 00 [000] r	CB	2	2	8	**Rotazione circolare a sinistra.**
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 [000] 110	CB	2	4	15	r / Reg. 000 B 001 C
RLC (IX+d)		↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 ← d → 00 000 110	DD CB	4	6	23	010 D 011 E 100 H 101 L 111 A
RLC(IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 ← d → 00 000 110	FD CB	4	6	23	

Gruppo di istruzioni di spostamento e rotazione

Mnemonico	Simbolo della operazione	7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	Codice macchina 76 543 210	HEX	N° dei bytes	N° dei cicli	N° dei periodi di clock	Commenti
RL s	CY ← 7←0 (ciclico)	↕	↕	X	0	X	P	0	↕	010					
	s ≡ r,(HL),(IX+d),(IY+d)														
RRC s	7→0 → CY (ciclico)	↕	↕	X	0	X	P	0	↕	001					
	s ≡ r,(HL),(IX+d),(IY+d)														
RR s	7→0 → CY (ciclico)	↕	↕	X	0	X	P	0	↕	011					
	s ≡ r,(HL),(IX+d),(IY+d)														
SLA s	CY ← 7←0 ← 0	↕	↕	X	0	X	P	0	↕	100					
	s ≡ r,(HL),(IX+d),(IY+d)														
SRA s	7→0 → CY	↕	↕	X	0	X	P	0	↕	101					
	s ≡ r,(HL),(IX+d),(IY+d)														
SRL s	0 → 7→0 → CY	↕	↕	X	0	X	P	0	↕	111					Rotazione del digit da sinistra a destra con l'accumulatore e HL.
	s ≡ r,(HL),(IX+d),(IY+d)														
RLD	A 7-4 3-0 7-4 3-0 (HL)	↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Il contenuto della metà alta dell'accumulatore non sarà influenzato.
RRD	A 7-4 3-0 7-4 3-0 (HL)	↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

• = Significa bit di stato non influenzato, 0 = Bit resettato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Il bit è influenzato secondo il risultato dell'operazione.

Gruppo di istruzioni di azzeramento, settaggio a 1 e test

Mnemonico	*Simbolo della operazione*	*Registro di stato* 7 S	6 Z	5	4 H	3	2 P/V	1 N	0 C	*Codice macchina* 76 543 210	HEX	*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*
BIT b, r	$Z \leftarrow r_b$	X	↕	X	1	X	X	0	●	11 001 011 01 b r	CB	2	2	8
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	X	↕	X	1	X	X	0	●	11 001 011 01 b 110	CB	2	3	12
BIT b, (IX+d)b	$Z \leftarrow \overline{(IX+d)}_b$	↕	X	i	X	X	0	●	●	11 001 011 11 001 011 ← d ← 01 b 110	DD CB	4	5	20
BIT b, (IY+d)b	$Z \leftarrow \overline{(IY+d)}_b$	X	↕	X	1	X	X	0	●	11 111 101 11 001 011 ← d ← 01 b 110	FD CB	4	5	20
SET b, r	$r_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 001 011 11 b r	CB	2	2	8
SET b, (HL)	$(HL)_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 001 011 11 b 110	CB	2	4	15
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 011 101 11 001 011 ← d ← 11 b 110	DD CB	4	6	23
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	●	●	X	●	X	●	●	●	11 111 101	FD	4	6	23

Commenti

r	Reg.
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	**Bit testato**
000	0 (Test)
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Gruppo di istruzioni di azzeramento, settaggio a 1 e test

Mnemonico	*Simbolo della operazione*	*Registro di stato*								*Codice macchina*		*N° dei bytes*	*N° dei cicli*	*N° dei periodi di clock*	*Commenti*
		7	6	5	4	3	2	1	0						
		S	Z		H		P/V	N	C	76 543 210	HEX				
RES b, s	Sb ← 0 s ≡ r, (HL) (IX+d), (IY+d)	•	•	X	•	X	•	•	•	11 001 011 ← d ← [11] b 110 10	CB				Il nuovo codice macchina sostituisce 11 di SET b,s con 10 e lo stato è influenzato nello stesso modo della istruzione SET.

La notazione Sb indica bit b (0-7) alla posizione S.
• = Significa bit di stato non influenzato, 0 = Bit resettato, 1 = Bit settato a 1, X = Bit indifferente (0 o 1), ↕ = Il bit è influenzato secondo il risultato dell'operazione.

Riassunto delle operazioni delle istruzioni dello Z-80 e loro relazione con il registro di stato.
(I commenti sono in inglese poichè sono correlazionati alle istruzioni mnemoniche)

Istruzione	Registro di stato								Commento
	D7 S	Z		H		P/V	N	D0 C	
ADD A, s; ADC A, s	↕	↕	X	↕	X	V	0	↕	8 bit add or add with carry
SUB, s; SBCA, s; CP, s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	↕	↕	X	1	X	P	0	0	Logical operations
OR s; XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	●	8-bit increment
DEC s	↕	↕	X	↕	X	V	1	●	8-bit decrement
ADD DD, SS	●	●	X	X	X	●	0	↕	16-bit add
ADC HL, SS	↕	↕	X	X	X	V	0	↕	16-bit add with carry
SBC HL, SS	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry
RLA; RLCA; RRA; RRCA	●	●	X	0	X	●	0	↕	Rotate accumulator
RL s; RLC s; RR s; RRC s; SLA s; SRA s; SRL s	↕	↕	X	0	X	P	0	↕	Rotate and shift locations
RLD; RRD	↕	↕	X	0	X	P	0	●	Rotate digit left and right
DAA	↕	↕	X	↕	X	P	●	↕	Decimal adjust accumulator
CPL	●	●	X	1	X	●	1	●	Complement accumulator
SCF	●	●	X	0	X	●	0	1	Set carry
CCF	●	●	X	X	X	●	0	↕	Complement carry
IN r, (C)	↕	↕	X	0	X	P	0	●	Input register indirect
INI; IND; OUTI; OUTD	X	↕	X	X	X	X	1	●	Block input and output
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	●	Z=0 if B≠0 otherwise Z=1
LDI; LDD	X	X	X	0	X	↕	0	●	Block transfer instructions
LDIR; LDDR	X	X	X	0	X	0	0	●	P/V=1 if BC≠0, otherwise P/V=0
CPI; CPIR; CPD; CPDR	X	↕	X	X	X	↕	1	●	Block search instructions Z=1 if A=(HL), otherwise Z=0 P/V=1 if BC≠0, otherwise P/V=0
LDA, I; LD A, R	↕	↕	X	0	X	IFF	0	●	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	X	↕	X	1	X	X	0	●	*The state of bit b of location s is copied into the Z flag*

Tabella I.-Tabella delle istruzioni del μP Z80.

certo incremento del numero delle istruzioni e della potenza delle stesse: cioè più grande è il numero delle istruzioni più specifica sarà l'operazione di ciascuna. Nelle tabelle sono esposte dettagliatamente le istruzioni dello Z80 in base al codice mnemonico. LD significa load o caricamento e consiste nel trasferire un dato da un registro all'altro o da una locazione di memoria a un registro. PUSH è utilizzata per trasferire il dato da un registro allo stack. POP è l'operazione inversa a PUSH. EX permette di scambiare i dati senza alterarli da un registro a un altro. CP è l'operazione di confronto. ADD è l'operazione di somma. SUB è l'operazione di sottrazione. AND è l'operazione logica dello stesso nome. OR è anch'essa l'operazione logica OR. XOR è l'operazione logica OR esclusivo. INC significa incrementare, DEC decrementare. DAA è un'operazione in BCD. CPL è un'operazione di complemento o inversione. NEG è un'operazione di complemento a due. CCF è un'operazione di complemento del bit di carry. SCF setta a uno il bit di carry. NOP è un'istruzione di non operare. HALT arresta il μP. DI, EI, IM0, IM1, IM2, sono istruzioni di trattamento degli interrupts. RL e RR sono operazioni di rotazione verso sinistra e verso destra rispettivamente. BIT è un'operazione di test. SET è un'operazione di settaggio a uno. RES è un'operazione di settaggio a zero. JP è un'operazione di salto. JR è un'operazione di salto relativo. DJNZ decrementa e salta se non è zero. CALL richiamo di subroutine. RET ritorno da subroutine. IN ingresso da una periferica. OUT uscita verso una periferica.

Tipi di indirizzamento

Il microprocessore dispone dei seguenti modi di indirizzamento:

- indirizzamento immediato: è utilizzato con istruzioni a due bytes, il primo è quello dell'istruzione, il secondo è l'operando o dato da trattare;
- indirizzamento immediato esteso: è utilizzato con istruzioni a tre bytes; l'operando occupa gli ultimi due;
- indirizzamento relativo: si utilizza per operare in prossimità dell'istruzione in corso ed è quindi in relazione con il contatore di programma; occupa due bytes;
- indirizzamento esteso: è utilizzato con istruzioni in cui si indica l'effettivo indirizzo; occupa tre bytes;
- indirizzamento indicizzato: l'indirizzo effettivo si ottiene sommando il contenuto dei registri indice con quello del terzo byte; necessita di tre bytes;
- indirizzamento tramite registri: è utilizzato per operazioni di

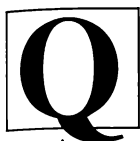
trattamento dei registri;
- indirizzamento indiretto: l'indirizzo effettivo si trova alla locazione di memoria indicata dall'istruzione.

Riassunto del significato delle notazioni utilizzate nella tabella

Le notazioni utilizzate nella tabella delle istruzioni del microprocessore Z80 sono le seguenti:

C = bit di carry;
Z = bit di risultato zero nell'accumulatore;
S = bit del segno del risultato nell'accumulatore;
P/V = bit di parità e overflow;
H = bit di carry di mezzo byte;
N = bit che indica il tipo dell'operazione precedente (somma o differenza);
• = il corrispondente bit non cambia nell'operazione;
0 = il bit corrispondente è stato resettato durante l'operazione eseguita;
1 = il corrispondente bit è stato settato durante l'operazione effettuata;
X = il valore del corrispondente bit è indifferente;
V = il bit corrispondente viene influenzato dal carry durante l'operazione effettuata;
P = il bit corrispondente viene influenzato dalla parità del risultato;
r = si riferisce a uno dei registri A, B, C, D, E, H, L;
s = si riferisce a qualsiasi locazione di memoria a cui si accede con otto bit;
ss = si riferisce a qualsiasi locazione di memoria a cui si accede con sedici bit;
R = si riferisce al contatore di refresh;
n = con questa lettera si indica un qualsiasi valore binario a otto bit, compreso tra 0 e 255 decimale;
nn = con queste lettere si indica un qualsiasi valore binario a sedici bit, compreso tra 0 e 65535 decimale.

CAPITOLO 6

MICROPROCESSORI A 8 BIT PIU' COMUNI (III)

Questo capitolo è dedicato a due microprocessori attualmente molto utilizzati come il 6800 della Motorola e il 6502 della Rockwell, che hanno caratteristiche molto simili.

Benchè prodotti da costruttori diversi, vengono considerati assieme per la loro similitudine di funzionamento e di programmazione.

Il microprocessore 6800

Il microprocessore 6800 venne messo in commercio dalla Motorola ed è attualmente uno dei più utilizzati, con altri della stessa serie che lo seguirono, come il 6802.

Le caratteristiche di questo µP sono: il 6800 è costruito in tecnologia NMOS, ha una capacità di indirizzamento di memoria di 64 Kbytes e frequenza che può variare dal MHz del 6800-A fino ai 2 MHz del 6800-B, prevede 72 istruzioni, e l'alimentazione è unica a +5 V, i modi di indirizzamento possono essere: diretto, relativo, immediato, indicizzato, esteso, o implicito. Relativamente agli interrupts, ne possiede uno logico (SWI) e due hardware (IRQ e NMI). Il suo bus dei dati è a otto bit.

Costituzione interna del 6800

In Fig. 1 si vedono i blocchi interni di questo microprocessore, dei quali si deve evidenziare come specifico, il blocco dei registri, mentre il resto è simile a quanto previsto generalmente in tutti i microprocessori.

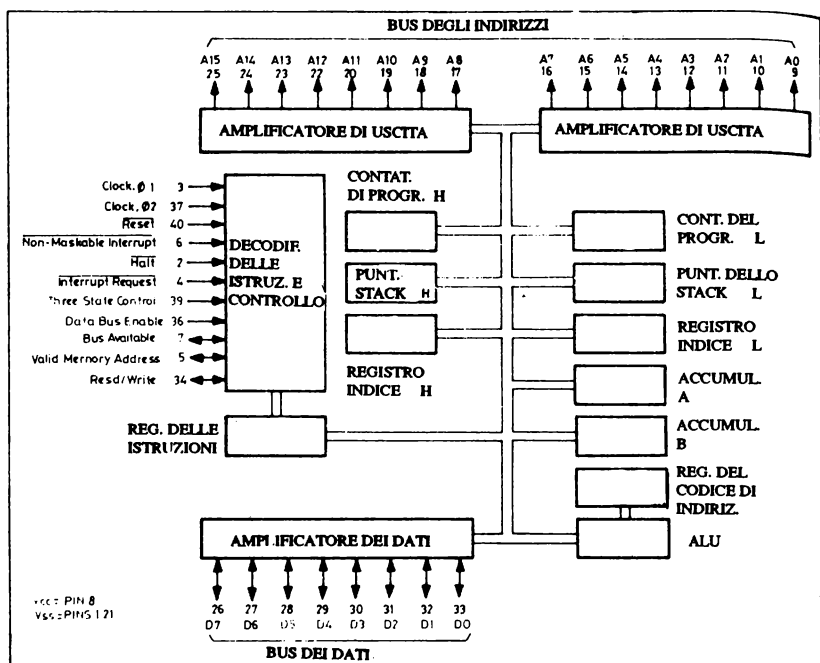



Fig. 1.-Diagramma interno a blocchi del microprocessore 6800.

Il 6800 dispone di sei registri interni ai quali farà riferimento il software.

L'uso di ogni registro è il seguente:

- accumulatori: il 6800 ha due registri accumulatori ai quali nelle istruzioni ci si riferisce come A e B. Ognuno di questi è da 8 bit, e attraverso di loro passano gli operandi diretti al µP ed i risultati forniti dall'ALU.
- registro indice: è un registro a due bytes, utilizzato come memoria temporanea per i dati o come registro indice per l'immagazzinamento dei dati in memoria con indirizzamento indicizzato.
- contatore di programma: è un registro a due bytes (16 bit) in cui viene memorizzato l'indirizzo di memoria dell'istruzione in corso.
- puntatore di stack: è un registro a due bytes in cui si memorizza

l'indirizzo della posizione di memoria nella quale è ubicato lo stack, in cui si memorizzano gli indirizzi di ritorno dalle subroutines o altri indirizzi o dati necessari. Questo stack può essere costituito da una memoria RAM, meglio se non volatile, che si può ottenere con una batteria tampone che rende l'alimentazione indipendente da quella del sistema.

- registro di stato: è un registro a otto bit, ognuno dei quali ha una propria funzione relativa allo stato funzionale del microprocessore. Il loro significato è:
  - — i bit 6 e 7 non vengono utilizzati e sono settati permanentemente a uno;
  - — il bit 5, detto H, serve per il carry di mezzo byte, il suo uso è indispensabile per le operazioni in BCD;
  - — il bit 4, detto I, è quello che permette gli interrupts mascherati, il permesso è concesso se tale bit è settato a uno;
  - — il bit 3, detto N, è quello che indica il segno del dato elaborato. Quando N=1, il dato è negativo (bit 7 del dato = 1);
  - — il bit 2, detto Z, indica quando il dato elaborato vale zero (quando ciò accade Z=1);
  - — il bit 1, detto V, indica, quando è settato a uno, che è avvenuto un overflow su qualche dato;
  - — il bit 0, detto C, è il bit di carry che si porta a uno quando l'operazione aritmetica eseguita ha prodotto un riporto.

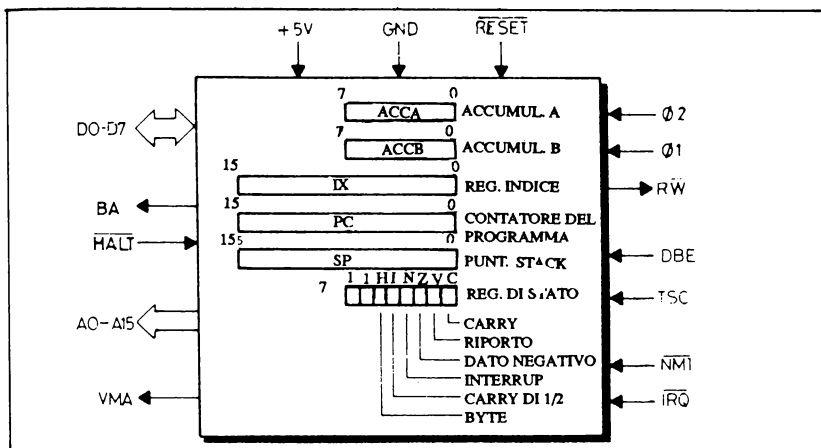



Fig. 2.-Dettaglio del blocco dei registri interni del μP 6800.

Descrizione dei terminali del 6800

Il microprocessore 6800 si trova in un contenitore a 40 piedini, le cui funzioni sono:

- A0-A15: è il bus degli indirizzi formato da 16 bit per poter indirizzare 64 Kbytes di memoria. Queste uscite possono essere collegate a circuiti TTL (Fan out=1).
- D0-D7: sono i terminali corrispondenti al bus dei dati, gli otto three state interni possiedono un fan out=1, per cui possono essere collegati a circuiti TTL con fan in=1.

  Si ricorda che il fan out è il numero di ingressi TTL con carico tipico di 1,6 mA (fan in) che si possono collegare in parallelo a uno di tali terminali.
- HALT: è un ingresso, resettando il quale si può arrestare il μP. Ciò accade solo dopo che è terminata l'istruzione in corso.
- TSC: è un ingresso mediante il quale si può portare ad alta impedenza il bus degli indirizzi ed il segnale di lettura- scrittura (R/W). Ciò accade quando TSC (Three State Control) = 1.
- R/W (Read/Write): è un'uscita, mediante la quale il μP indica alla memoria e alle periferiche se l'operazione è di lettura (livello logico uno) o scrittura (livello logico zero).
- VMA (Valid Memory Address): questa uscita indica in quale istante l'indirizzo è valido. E' effettivamente un impulso positivo di sincronismo che serve per indicare il momento in cui si può prendere l'indirizzo dal bus, dopo che è trascorso il tempo critico dei fronti.
- DBE (Data Bus Enable): è un ingresso attraverso il quale si controlla il bus dei dati, portandolo ad alta impedenza o consentendo il flusso dei dati.
- BA (Bus Available): è un'uscita che, quando viene settata a uno, indica all'esterno che in quel momento il μP si è messo in attesa liberando i bus.

  Questa uscita si attiva quando il μP ha ricevuto un HALT, o quando si trova in stato di attesa in seguito all'esecuzione dell'istruzione WAIT.
- IRQ (Interrupt Request): mediante questo ingresso le periferiche possono richiedere un interrupt al microprocessore, il quale completa l'istruzione che sta eseguendo in tale istante, e salta ad

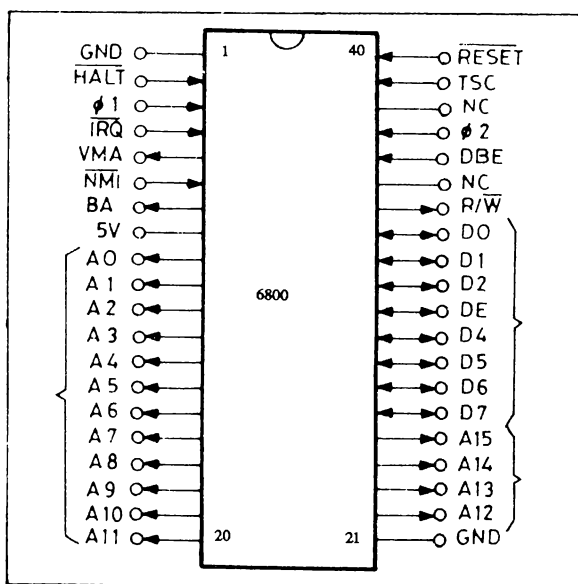



Fig. 3.-Terminali del microprocessore 6800 e loro funzioni.

un programma interno di trattamento degli interrupts.

L'ingresso HALT deve essere settato a uno perchè gli interrupts siano permessi. L'ingresso IRQ necessita di una resistenza esterna da 3 KΩ collegata a +5 V perchè tutti gli ingressi di richiesta di interrupts possano essere collegati a questo terminale, formando un *OR* cablato.

- RESET: è l'ingresso di azzeramento, utilizzato per azzerare e inizializzare il microprocessore dopo averlo alimentato, o quando si vuole reinizializzarlo.

  Questo segnale avvia una routine interna al microprocessore che obbliga ad iniziare l'esecuzione del programma principale.

  In tale istante sul bus degli indirizzi comparirà l'indirizzo FFFE, locazione di memoria che, con la successiva FFFF, contiene l'indirizzo da cui inizia il programma principale, poichè il contenuto di queste locazioni sarà fornito al contatore di programma (PC) come primo indirizzo di esecuzione. La routine interna di inizializzazione resetta anche il bit di interrupt dello status per bloccare tutte le interruzioni durante l'inizializzazione.

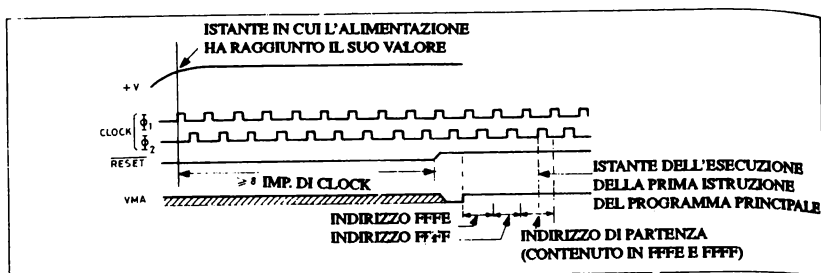



Fig. 4.-Diagramma dei tempi dei segnali che intervengono nell'azzeramento e nell'inizializzazione del microprocessore. L'indirizzo iniziale del programma principale è contenuto alle locazioni FFFE e FFFF.

In Fig. 4 si vede il diagramma dei tempi dei segnali che intervengono in questa fase.

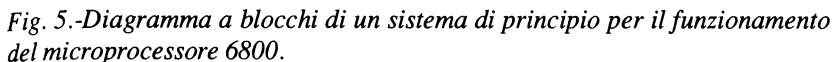
- NMI (Not Maskable Interrupt): è un ingresso attraverso il quale si richiede un interrupt non mascherabile. Quando questo ingresso viene resettato a zero, il µP completa l'istruzione in corso e soddisfa la richiesta della periferica.

Hardware di base per il 6800

In Fig. 5 è indicato lo schema a blocchi di un semplice sistema da aggiungere al microprocessore 6800 perchè possa eseguire un programma.

Fondamentalmente comprende il microprocessore pilotato dalle due fasi di clock fornite dal circuito integrato 6871, un blocco con i circuiti di inizializzazione per ottenere l'inizio dell'esecuzione del programma dopo aver fornito l'alimentazione, il blocco di memoria in cui una ROM o una EPROM contiene il programma, e una RAM per l'elaborazione dei dati e delle variabili. L'ultimo blocco è costituito da una PIA, o circuito che agisce da interfaccia tra il microprocessore e le periferiche.

Tutti questi blocchi sono collegati al µP mediante i bus dei dati, degli indirizzi e di controllo.

Fig. 5.-Diagramma a blocchi di un sistema di principio per il funzionamento del microprocessore 6800.

La PIA MC-6820

Il circuito integrato PIA MC-6820 fornisce al µP 6800 un metodo flessibile per il suo collegamento alle periferiche. Benchè sia internamente un circuito relativamente complesso, permette una certa varietà di collegamenti con la CPU mediante un minimo di circuiteria logica aggiuntiva, ed una semplificazione della programmazione.

Attualmente quasi tutti i microprocessori dispongono di circuiti associati per facilitare i collegamenti della CPU con l'esterno. Questi circuiti sono integrati di tipo LSI in contenitori a quaranta piedini, ai quali deve essere collegato il µP da un lato, e le periferiche dall'altro. Sono interfacce complete denominate PIA (Peripheral Interface Adapter). La Motorola costruttrice del µP 6800 e della PIA MC-6820 è stata la prima ad introdurre questo tipo di circuito nei suoi sistemi; in seguito diversi fabbricanti hanno seguito tale esempio. La PIA MC-6820 è costituita da due blocchi di accesso al bus dei dati, il blocco A e il blocco B, entrambi a otto bit. Ognuno di questi bit viene collegato internamente al corrispondente bit del bus dei dati del µP, e può essere trasformato in ingresso o uscita, in base al contenuto precedentemente caricato

nel registro interno DDRA.

Quando un bit di tale registro è settato a uno, il corrispondente bit del blocco di accesso A diventa di uscita, al contrario, quando tale bit e resettato, l'omologo del blocco di accesso A diventa di ingresso. Ciò permette un controllo indipendente di ognuno degli otto bit del blocco di accesso A, e B (DDRA controlla il blocco A e DDRB controlla il blocco B), il flusso di dati tra il bus dei dati ed i blocchi passa attraverso i registri ORA e ORB, mentre DDRA e DDRB indicano il senso di trasferimento.

Infine i registri CRA e CRB sono registri di controllo, monitorizzano gli interrupts tra CPU e periferiche, e indirizzano i registri dei dati e il loro trasferimento.

Il bus di controllo della PIA è costituito da CS0, CS1 e CS2, che servono per indirizzarla. RS0 e RS1 servono per selezionare il blocco A o B, R/W per interpretare l'operazione di lettura o scrittura del µP, Reset per azzerare il contenuto di tutti i registri interni, mentre Enable è collegato al clock del sistema.

Per comprendere meglio quanto esposto, si faccia riferimento alla Fig. 6.

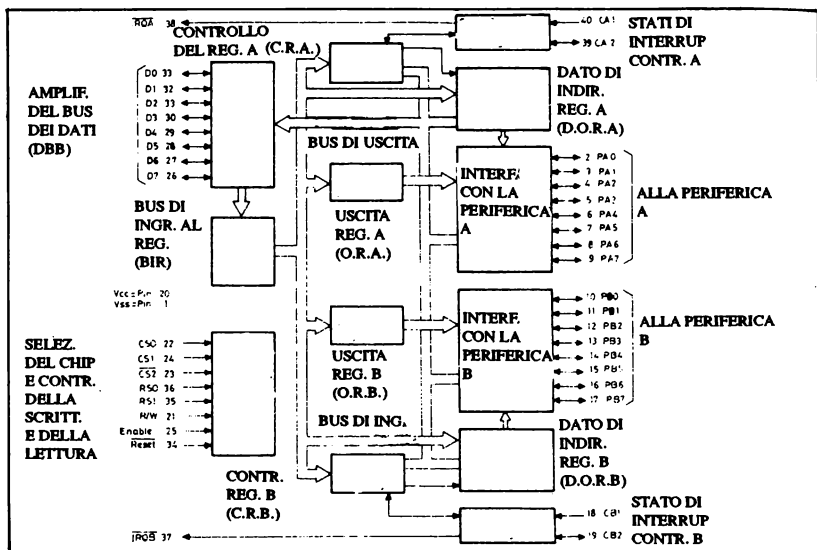



Fig. 6.-Blocchi interni della PIA MC-6820, collegabile al microprocessore 6800.

Modi di indirizzamento del 6800

Il microprocessore 6800 dispone dei seguenti modi di indirizzamento:

- *immmediato:* in questo tipo di indirizzamento il dato o operando dell'istruzione è fornito dopo il byte del codice macchina;
- *diretto:* questo indirizzamento si utilizza quando il dato si trova in una locazione di memoria indicata, a partire dall'inizio della pagina, dal secondo byte, perciò questo indirizzamento può solo arrivare a 256 posizioni a partire dall'inizio di una pagina. Viene anche detto paginato;
- *indicizzato:* il contenuto del secondo byte viene sommato a quello del registro indice per ottenere l'indirizzo finale, ma il contenuto di tale registro non si altera poichè il risultato viene salvato in un registro ausiliario a 16 bit;
- *esteso:* il secondo e il terzo byte determinano l'indirizzo assoluto su cui opererà la corrispondente istruzione;
- *implicito o inerente:* il dato o operando è implicito nell'istruzione;
- *relativo:* il contenuto del secondo byte dell'istruzione viene sommato al contenuto del registro contatore di programma per ottenere l'indirizzo finale. Si usa nelle istruzioni di salto.

Interrupts del microprocessore 6800

Gli interrupts nel µP 6800 sono trattati mediante i segnali introdotti via hardware sui terminali NMI e IRQ nel modo seguente: quando viene attivato il segnale NMI: resettandolo a basso livello, inizia una sequenza di *interrupt non mascherabile,* il che significa che l'interrupt viene sempre trattato, indipendentemente dal valore del bit 1 del registro di stato.

Questo interrupt provoca il caricamento del PC con il contenuto del vettore memoria, costituito dagli indirizzi FFFC e FFFD, in cui è memorizzato l'indirizzo iniziale della routine di trattamento di tale interrupt.

Quando il segnale IRQ viene attivato dall'esterno, sempre mediante un livello basso di tensione, viene eseguita una sequenza di *interrupt mascherabile,* cioè l'interrupt verrà eseguito solo se il bit 1 del registro di stato è a livello zero.

Il vettore memoria per il trattamento di tale interrupt è localizzato alle posizioni FFF8 e FFF9.

ISTRUZIONI DEL REGISTRO INDICE E DELLO STACK

Operazioni	*Mnemonico*	*Modo di indirizzamento*															*Commenti*	*Registro di stato*					
		Immediato			*Diretto*			*Indicizzato*			*Esteso*			*Implicito*				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	8C	5	3				X_H-M.X_L-(M+1)	●	●	7	↕	7	●
Decrement Index Reg	DEX													09	4	1	X - 1 → X	●	●	●	↕	●	●
Decrement Stack Pnts	DES													34	4	1	SP - 1 → SP	●	●	●	●	●	●
Increment Index Reg	INX													03	4	1	X + 1 → X	●	●	●	↕	●	●
Increment Stack Part	INS													31	4	1	SP + 1 → SP	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				M → X_H.(M+1) → X_L	●	●	9	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				M → SP_H (M+1) → SP_L	●	●	9	↕	R	●
Store Index Reg	STX				OF	5	2	EF	7	2	FF	6	3				X_H → M.X_L → (M+1)	●	●	9	↕	R	●
Store Stack Patr	STS				9F	5	2	AF	7	2	BF	6	3				SP_H → M.SP_L → (M+1)	●	●	9	↕	R	●
Indx Reg → Stack Pntr	TXS													35	4	1	X - 1 → SP	●	●	●	●	●	●
Stack Pntr → Indx Reg	TSX													30	4	1	SP + 1 → X	●	●	●	●	●	●

SIGNIFICATO DEI SIMBOLI UTILIZZATI

OP = Codice di operazione. ~ = Numero di cicli. # = Numero di bytes. + = Somma. - = Differenza. • = Funzione logica AND.
M-SP = Posizione di memoria mediante il puntatore di stack. + = Funzione logica OR. + = Funzione logica XOR. M = Complemento di M. → =
Trasferimento logico.
0 = Bit = 0. 00 = Byte = 0.

SIMBOLI DEL REGISTRO DI STATO

H = Carry di mezzo byte. I = Interrupt mascherato. N = segno negativo. Z = dato uguale a zero. V = Overflow. C= Carry. R = Reset. S = Set.
↕ = Test e set se la condizione è verificata. • = Bit non influenzato.

TABELLA DELLE ISTRUZIONI DEL μP 6800

Istruzioni dell'accumulatore e della memoria

Operazioni	Mnemonico	Modo di indirizzamento															Commenti	Registro di stato					
		Immediato			Diretto			Indicizzato			Esteso			Implicito				8	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	9B	2	2	9B	3	2	AB	5	2	BB	4	3				A+M → A	↕	●	↕	↕	↕	↕
	ADDB	DB	2	2	DB	3	2	EB	5	2	FB	4	3				B+M → B	↕	●	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A+B → A	↕	●	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A+M+C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B+M+C → B	↕	●	↕	↕	↕	↕
And	ANDA	B4	2	2	94	3	2	A4	5	2	B4	4	3				A●M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B●N → B	●	●	↕	↕	R	●
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A●M	●	●	↕	↕	R	●
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				8●M	●	●	↕	↕	R	●
Clear	CLR							6F	7	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A-M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				8-M	●	●	↕	↕	↕	↕
Compare Acmltr	CBA													51	2	1	A-B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M → M	●	●	↕	↕	R	S
	COMA													43	2	1	A → A	●	●	↕	↕	R	S
	COMB													53	2	1	8 → B	●	●	↕	↕	R	S
Complement, 2's	NEG							80	7	2	70	6	3				00-M → M	●	●	↕	↕	1	2
(Negate)	NEGA													40	2	1	00-A → A	●	●	↕	↕	1	2
	NEGB													50	2	1	00-B → B	●	●	↕	↕	1	2
Decimal Adjust, A	DAA													19	2	1	CONVERTIRE I DATI BINARI IN BCD	●	●	↕	↕	↕	3
Decrement	DEC							6A	7	2	7A	6	3				M-1 → M	●	●	↕	↕	4	●
	DECA													4A	2	1	A-1 → A	●	●	↕	↕	4	●
	DECB													5A	2	1	B-1 → B	●	●	↕	↕	4	●

Operazioni	Mnemonico	Modo di indirizzamento: Immediato			Diretto			Indicizzato			Esteso			Implicito			Commenti	Registro di stato: 8	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	88	4	3				A+M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				8+M → B	●	●	↕	↕	R	●
Increment	INC							6C	7	2	7C	6	3				M+1 → M	●	●	↕	↕	5	●
	INCA													4C	2	1	A+ → A	●	●	↕	↕	5	●
	INCB													5C	2	1	B+1 → B	●	●	↕	↕	5	●
Load Acmitr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	06	3	2	E6	5	2	F6	4	3				M → B	●	●	↕	↕	R	●
Or Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A+M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B+M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	4	1	A → MSP.SP - 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → MSP.SP - 1 → SP	●	●	●	●	●	●
Pull Oata	PULA													32	4	1	SP+1 → SP.MSP → A	●	●	●	●	●	●
	PULB													33	4	1	SP+1 → SP.MSP → B	●	●	●	●	●	●
Rotate Left	ROL							69	7	2	79	6	3				M	●	●	↕	↕	6	↕
	ROLA													49	2	1	A ←	●	●	↕	↕	6	↕
	ROLB													59	2	1	B C b7 ← b0	●	●	↕	↕	6	↕
Rotate Right	ROR							66	7	2	76	6	3				M	●	●	↕	↕	6	↕
	RORA													46	2	1	A →	●	●	↕	↕	6	↕
	RORB													56	2	1	B C b7 → b0	●	●	↕	↕	6	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M ←	●	●	↕	↕	6	↕
	ASLA													48	2	1	A ← ← 0	●	●	↕	↕	6	↕
	ASLB													58	2	1	B C b7 b0	●	●	↕	↕	6	↕
Shift Right. Arithmetic	ASR							67	7	2	77	6	3				M →	●	●	↕	↕	6	↕
	ASRA													47	2	1	A →	●	●	↕	↕	6	↕
	ASRB													57	2	1	B b7 b0 C	●	●	↕	↕	6	↕
Shift Right. Logic	LSR							64	7	2	74	6	3				M →	●	●	R	↕	6	↕
	LSRA													44	2	1	A 0 →	●	●	R	↕	6	↕
	LSRB													54	2	1	B B7 b7 C	●	●	R	↕	6	↕

Operazioni	Mnemonico	Modo di indirizzamento															Commenti	Registro di stato					
		Immediato			Diretto			Indicizzato			Esteso			Implicito				8	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Store Acmftr	STAA				97	4	2	A7	6	2	B7	5	3				A → M	●	●	↕	↕	R	●
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A - M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B - M → B	●	●	↕	↕	↕	↕
Subtract Acmltrs	SBA													10	2	1	A - B → A	●	●	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A - M - C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B - M - C → B	●	●	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test. Zero or Minus	TST							6D	7	2	7D	6	3				M - 00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A - 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B - 00	●	●	↕	↕	R	R

Istruzioni di salto

Operazioni	*Mnemonico*	*Modo di indirizzamento*												*Commenti*	*Registro di stato*					
		Immediato			*Indicizzato*			*Esteso*			*Implicito*				8	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	4	2										C=0	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	4	2										C=1	●	●	●	●	●	●
Branch If Carry Set	BCS	25	4	2										Z=1	●	●	●	●	●	●
Branch If=Zero	BEQ	27	4	2										$N \oplus V=0$	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	4	2										$Z+(N \oplus V)=0$	●	●	●	●	●	●
Branch If>Zero	BGT	2E	4	2										C+Z=0	●	●	●	●	●	●
Branch If Higher	BHI	22	4	2										$Z+(N \oplus V)=1$	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	4	2										C+Z=1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	4	2										$N \oplus V=1$	●	●	●	●	●	●
Branch If < Zero	BLT	2D	4	2										N=1	●	●	●	●	●	●
Branch If Minus	BMI	2B	4	2										Z=0	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	4	2										V=0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	4	2										V=1	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	4	2										N=0	●	●	●	●	●	●
Branch If Plus	BPL	2A	4	2											●	●	●	●	●	●
Branch To Subroutine	BSR	8D	8	2											●	●	●	●	●	●
Jump	JMP				6E	4	2	7E	3	3				Ved. operazioni speciali	●	●	●	●	●	●
Jump To Subroutine	JSR				AD	8	2	BD	9	3					●	●	●	●	●	●
No Operation	NOP										02	2	1	Avanza solo il PC	●	●	●	●	●	●
Return From Interrupt	RTI										3B	10	1					10		
Return From Subroutine	RTS										39	5	1	Ved. operazioni speciali	●	●	●	●	●	●
Software Interrupt	SWI										3F	12	1		●	●	●	●	●	●
Wait for interrupt	WAI										3E	9	1		●	11	●	●	●	●

Significato dei simboli utilizzati

OP = Codice di operazione
~ = Numero di cicli
# = Numero di bytes.
+ = Somma
- = Differenza
• = Funzione logica AND
M-SP = Posizione di memoria mediante il puntatore di stack

+ = Funzione logica OR
$\oplus$ = Funzione logica XOR
M = Complemento di M
→ = Trasferimento verso.
0 = Bit = 0
00 = Byte = 0

NOTA: Le istruzioni con indirizzamento dell'accumulatore sono incluse nella colonna di indirizzamento implicito.

Istruzioni di manipolazione del registro di stato

Operazioni	*Mnemonico*	*Indirizzamento*			*Commenti*	*Registro di stato*					
		Implicito				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR			12			
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

Note al registro di stato

1 (Bit V) test: risultato = 10000000?
2 (Bit C) test: risultato = 00000000?
3 (Bit C) test: il valore decimale del carattere BCD è maggiore di 9?
4 (Bit V) test: l'operando = 10000000 maggiore all'esecuzione?
5 (Bit V) test: l'operando = 01111111 maggiore all'esecuzione?
6 (Bit V) test: lo pone uguale al risultato di NOC dopo lo spiazzamento
7 (Bit N) test: il bit di segno = 1?
8 (Bit N) test: complemento a due della sottrazione?
9 (Bit N) test: risultato minore di zero? (Bit 15 = 1)
10 (Tutti) carica il registro di stato dallo stack
11 (Bit I) selezionato quando avviene un interrupt. Se è stato preventivamente effettuato un interrupt non mascherato richiede lo stato di attesa
12 (Tutti) Selezionati secondo il contenuto dell'accumulatore A. (Il bit selezionato è settato a uno se la condizione è vera e resettato in caso contrario.)

Simboli del registro di stato

H = Carry di mezzo byte
I = Interrupt mascherato
N = Segno negativo
Z = Dato uguale a zero
V = Overflow
C = Carry
R = Reset (settaggio a zero)
S = Set (settaggio a uno)
• = Bit non influenzato
↕ = Test e settaggio se la condizione è vera.

Tabella delle istruzioni del microprocessore 6800, in funzione del codice macchina. In essa compaiono i modi di indirizzamento.
I due digit esadecimali si prendono in ordine. Esempio: 4A = DEC A (decrementa l'accumulatore A).

I \ J	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		NOP inh					TAP inh	TPA inh	INX inh	DEX inh	CLV inh	SEV inh	CLC inh	SEC inh	CLI inh	SEI inh
1	SBA	CBA					TAB inh	TBA inh		DAA inh		ABA inh				
2	BRA rel		BHI rel	BLS rel	BCC rel	BCS rel	BNE rel	BEQ rel	BNC rel	BNS rel	BPL rel	BMI rel	BGE rel	BLT rel	BGT rel	BLE rel
3	ISX inh	INS inh	PUL A	PUL B	DES inh	TXS inh	PSH A	PSH B		PTS inh		RTI inh			WAI inh	SWI inh
4	NEGA			COMA	LSR A		ROR A	ASR A	ASL A	ROL A	DEC A		INC A	TST A		CLR A
5	NEGB			COMB	LSR B		ROR B	ASR B	ASL B	ROL B	DEC B		INC B	TST B		CLR B
6	NEG ind			COM ind	LSR ind		ROR ind	ASR ind	ASL ind	RCL ind	DEC ind		INC ind	TST ind	JMP ind	CLR ind
7	NEG ete			COM ete	LSR ete		ROR ete	ASR ete	ASL ete	ROL ete	DEC ete		INC ete	TST ete	JMP ete	CLB ete
8	SUB A imm	CMP A imm	SBC A imm		ANDA imm	BIT A imm	LDA A imm		EOR A imm	ADC A imm	ORA A imm	ADDA imm	CPX A imm	BSR rel	LDS imm	
9	SUB A dir	CPM A dir	SBC A dir		ANDA dir	BIT A dir	LDA A dir	STA A dir	EOR A dir	ACD A dir	ORA A dir	ADDA dir	CPX A dir		LDS dir	STS dir
A	SUB A ind	CMP A ind	SBC A ind		ANDA ind	BIT A ind	LDA A ind	STA A ind	EOR A ind	ADC A ind	ORA A ind	ADDA ind	CPX A ind	JSR ind	LDS ete	STS ind
B	SUB A ete	CMP A ete	SBC A ete		ANDA ete	BIT A ete	LDA A ete	STA A ete	EOR A ete	ADC A ete	ORA A ete	ADDA ete	CPX A ete	JSR ete	LDS ind	STS ete
C	SUB B imm	CMP B imm	SBC B imm		ANDB imm	BIT B imm	LDA B imm		EOR B imm	ADC B imm	ORA B imm	ADDB imm			LDX imm	
D	SUB B dir	CMP B dir	SBC B dir		ANDB dir	BIT B dir	LDA B dir	STA B dir	EOR B dir	ADC B dir	ORA B dir	ADDB dir			LDX B dir	STX B dir
E	SUB B ind	CMP B ind	SBC B ind		ANDB ind	BIT B ind	LDA B ind	STA B ind	EOR B ind	ACD B ind	ORA B ind	ADDB ind			LDX ind	STX ind
F	SUB B ete	CMP B ete	SBC B ete		ANDB ete	BIT B ete	LDA B ete	STA B ete	EOR B ete	ADC B ete	ORA B ete	ADDB ete			LDX ete	STX ete

BSR: ISTRUZIONI DI CHIAMATA A SUBROUTINE
PC
Programma principale
n 8D=BSR
n+1 ±K=SEL
n+2 Istr. seguente
→SP Stack
SP-2
SP-1 (n+2)H
sp (n+2)L
PC
n+2±K
Subroutine
1° Istr.
K = Dato a 8 bits n+2 formato da (n+2)
6 (ae+2)L
JMP: ISTRUZIONI DI SALTO
Indirizzamento indicizzato
PC
Programma principale
n GE=JMP
n+1 K=SEL
X+K Istr. Succ.
Indirizzamento esteso
PC
Programma principale
n 7E=JMP
n+1 KH= Istr. Succ.
n+2 KL= Istr. Succ.
K Istr. Succ.
RST: ISTRUZIONI DI RIENTRO DA SUBROUTINE
PC
S
Subroutine
39=RTS
SP
⇒SP
SP+1 NH
→SP+2 NL
Stack
PC
n
Programma principale
Istr. Succ.
RTI: ISTRUZIONI DI RITORNO CON INTERRUPT
PC
S
Programma Interrupt
3B=RTI
SP
Stack
SP Codice di condizione
SP+1 Accumulatore B
SP+2 Accumulatore A
SP+3 Reg. Indice (X4) (X4)
SP+4 Reg. Indice (XL) (XL)
SP+5 NH
SP+6 NL
SP+7
PC
n
Programma principale
Istr. Succ.

Tabella del tipo di indirizzamento di ogni istruzione e suo tempo di esecuzione in microcicli

	Doppio operando	Accumul. X	Immediato	Diretto	Esteso	Indicizzato	Implicito	Relativo
ABA		●	●	●	●	●	2	●
ADC	X	●	2	3	4	5	●	●
ADD	X	●	2	3	4	5	●	●
AND	X	●	2	3	4	5	●	●
ASL		2	●	●	6	7	●	●
ASR		2	●	●	6	7	●	●
BCC		●	●	●	●	●	●	4
BCS		●	●	●	●	●	●	4
BEA		●	●	●	●	●	●	4
BGE		●	●	●	●	●	●	4
BGT		●	●	●	●	●	●	4
BHI		●	●	●	●	●	●	4
BIT	X	●	2	3	4	5	●	●
BLE		●	●	●	●	●	●	4
BLS		●	●	●	●	●	●	4
BLT		●	●	●	●	●	●	4
BMI		●	●	●	●	●	●	4
BNE		●	●	●	●	●	●	4
BPL		●	●	●	●	●	●	4
BRA		●	●	●	●	●	●	4
BSR		●	●	●	●	●	●	8
BVC		●	●	●	●	●	●	4
BVS		●	●	●	●	●	●	4
CBA		●	●	●	●	●	2	●

	Doppio operando	Accumul. X	Immediato	Diretto	Esteso	Indicizzato	Implicito	Relativo
CLC		●	●	●	●	●	2	●
CLI		●	●	●	●	●	2	●
CLR		2	●	●	6	7	●	●
CLV		●	●	●	●	●	2	●
CMP	X	●	2	3	4	5	●	●
COM		2	●	●	6	7	●	●
CPX		●	3	4	5	6	●	●
DAA		●	●	●	●	●	2	●
DEC		2	●	●	6	7	●	●
DES		●	●	●	●	●	4	●
DEX		●	●	●	●	●	4	●
EOR	X	●	2	3	4	5	●	●
INC		2	●	●	6	7	●	
INS		●	●	●	●	●	4	
INX		●	●	●	●	●	4	
JMP		●	●	●	3	4	●	
JSR		●	●	●	9	8	●	
LDA	X	●	2	3	4	5	●	
LDS		●	3	4	5	6	●	
LDX		●	3	4	5	6	●	
LSR		2	●	●	6	7	●	
NEG		2	●	●	6	7	●	
NOP		●	●	●	●	●	2	
ORA	X	●	2	3	4	5	●	

	Doppio operando	Accumul. X	Immediato	Diretto	Esteso	Indicizzato	Implicito	Relativo
PSH		●	●	●	●	●	4	
PUL		●	●	●	●	●	4	
ROL		2	●	●	6	7	●	
ROR		●	●	●	6	7	●	
RTI		●	●	●	●	●	10	
RTS		●	●	●	●	●	5	
SBA		●	●	●	●	●	2	
SBC	X	●	2	3	4	5	●	
SEC		●	●	●	●	●	2	
SEI		●	●	●	●	●	2	
SEV		●	●	●	●	●	2	
STA	X	●	●	4	5	6	●	
STS		●	●	5	6	7	●	
STX		●	●	5	6	7	●	
SUB	X	●	2	3	4	5	●	
SWI		●	●	●	●	●	12	
TAB		●	●	●	●	●	2	
TAP		●	●	●	●	●	2	
TBA		●	●	●	●	●	2	
TPA		●	●	●	●	●	2	
TST		2	●	●	6	7	●	
TSX		●	●	●	●	●	4	
TSX		●	●	●	●	●	4	
WAI		●	●	●	●	●	9	

Tabella del significato delle istruzioni

ABA	Somma di accumulatori
ADD	Somma con carry
ADC	Somma
AND	Funzione logica And
ASL	Spostamento a sinistra
BRA	Salto sempre
BSR	Salto a subroutine
BVC	Salto se non c'è overflow
BVS	Salto se c'è overflow
CBA	Confronto tra accumulatori
CLC	Azzera carry
CLI	Cancella interrupt mascherato
CLR	Azzera
CLV	Azzera overflow
CMP	Confronto
COM	Complemento
CPX	Confronto con il registro indice
DAA	Regolazione decimale
DEC	Decremento
DES	Incrementa il puntatore di stack
DEX	Decrementa il registro indice
EOR	Funzione logica or esclusivo
INC	Incrementa
INS	Incrementa il puntatore di stack
ASR	Spostamento verso destra
BCC	Salto se non c'è carry
BCS	Salto e c'è carry
BEQ	Salto se diverso da zero
BGE	Salto se maggiore o uguale zero
BGT	Salto se è maggiore di zero
BHI	Salto se superiore
INX	Incrementa il registro indice
JMP	Salto
JSR	Salto a subroutine
LDA	Carica l'accumulatore
LDS	Carica il puntatore di stack
LDX	Carica il registro indice
LSR	Rotazione a destra
NEG	Negazione
NOP	Nessuna operazione
ORA	Funzione or con l'accumulatore
PSH	Inserisce dato in stack
PUL	Estrae dato da stack
ROL	Rotazione verso sinistra
ROR	Rotazione verso destra
RTI	Ritorno da sub. di interrupt
RTS	Ritono da subroutine
SBA	Sottrazione con l'accumulatore
SBC	Sottrazione con carry
SEC	Selezione di carry
BIT	Test del bit
BLE	Salto se minore o uguale
BLS	Salto se inferiore a dato
BLT	Salto se minore di zero
BMI	Salto se minore
BNE	Salto se diverso da zero
BPL	Salto se maggiore
SEI	Selezione interrupt mascherato
SEV	Selezione dell'overflow
STA	Memorizzazione nel-l'accumulatore
STS	Memorizzazione del registro di stack
STX	Memorizzazione registro indice
SUB	Sottrazione
SWI	Interrupt via software
TAB	Trasferimento tra accumulatori
TAP	Trasferimento degli accumulatori allo status
TBA	Trasferimento tra accumulatori
TPA	Trasferimento dallo status agli accumulatori
TST	Test
TSX	Trasferimento del puntatore di stack al registro indice
TXS	Trasferimento del registro indice al puntatore di stack
WAI	Attesa di interrupt

Repertorio delle istruzioni del 6800 suddivise per operazioni

1. Operazioni sui registri a 8 bit

A.	Operazioni aritmetiche a 2 operandi	ABC ADC ADD SBA SBC SUB
B.	Operazioni aritmetiche a 1 operando	CLR DAA DEC INC NEG
C.	Confronti e tests (Prove)	CBA CMP TST
D.	Scorrimento e rotazione	ASL ASR LSR ROL ROR
E.	Funzioni logiche	AND BIT COM EOR ORA
F.	Caricamento e memorizzazione	LDA STA PSH PUL
G.	Trasferimenti	TAB TBA

2.Controllo dei salti

A.	Salti condizionati	BCC BCS BEQ BGE BGT BHI BLE BLS BLT BMI BNE BPL BVC BVS
B.	Salti incondizionati	BRA NOP JMP
C.	Controlli di subroutines	BSR JSR RTS
D.	Controlli di interrupts	RTI SWI WAI

3. Controlli del registro indice e del puntatore di stack

A.	Registro indice	DEX INX LDX STX CPX
B.	Puntatore di stack	DES INS LDS STS
C.	Trasferimenti	TSX TXS

4. Controllo del registro di stato

A.	Controllo di bit	CLC CLI CLV SEC SEI SEV
B.	Trasferimento di byte	TAP TPA

5. Specifiche complessive — END EQU FCB FCC FDB MON NAM OPT ORG PAGE RMB SPC

Il microprocessore 6802

Questo microprocessore venne commercializzato dalla Motorola come variante del 6800 dopo i vari progressi nel campo dell'integrazione. In effetti il 6802 comprende nello stesso CHIP i circuiti completi del 6800, il circuito di clock e 128 bytes di memoria RAM.

In tal modo il 6802 è completamente compatibile, per quanto riguarda il software con il 6800.

Il microprocessore 6802 si presenta incapsulato nel classico contenitore a quaranta piedini. La distribuzione dei segnali sugli stessi è la stessa del 6800, tranne per quattro terminali che cambiano denominazione e che sono:

- XTAL 1 e XTAL 2, che nel 6802 costituiscono i collegamenti esterni per il cristallo di quarzo che regola la frequenza del generatore di clock interno su quella propria del quarzo (nel 6800 il generatore del clock è completamente esterno); nel 6800 XTAL 1 è occupato da TSC (controllo del three state) e XTAL 2 da un terminale non collegato.
- MR, che è un terminale per collegare il µP a memorie e periferiche lente; nel 6800 questo terminale è occupato dall'ingresso Φ1 del clock esterno.
- ME, terminale tramite cui si accede alla memoria RAM interna;

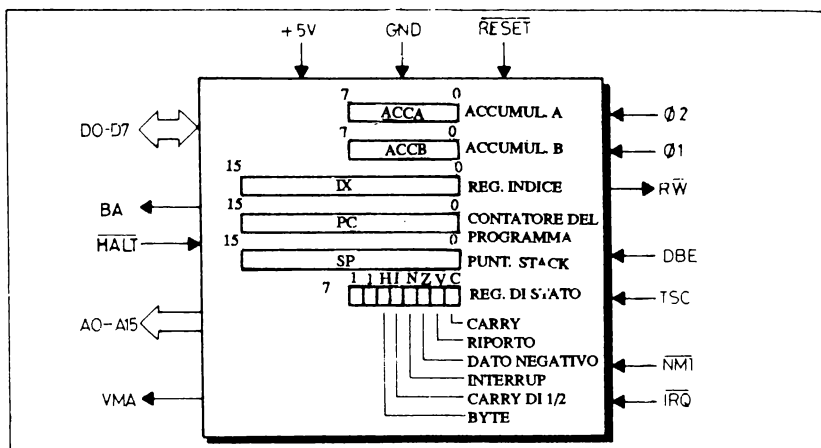



Fig. 7.-Dettaglio del blocco dei registri interni e della Ram a 128 bytes, integrati nel µP 6802.

è l'ingresso che permette di leggere o scrivere in questa RAM. Nel 6800 questo terminale è occupato da DBE che controlla il bus dei dati.

- VCC-S, terminale che permette di preservare la memoria interna da black-out dell'alimentazione, collegandogli una tensione esterna a 5 V, fornita da pile o batterie che lo alimentino in qualsiasi caso.

L'inserimento di memoria RAM o PROM nello stesso CHIP del microprocessore, è una tendenza degli ultimi progetti, in quanto in tal modo, il CHIP del microprocessore si trasforma in una completa CPU per applicazioni semplici.

Per quanto riguarda la struttura interna del µP 6802, questo è costituito da un blocco da 128 bytes o ottetti indirizzabili in esadecimale da 0000 a 007F, con la particolarità che i primi 32 sono non volatili e possono essere salvaguardati con il metodo descritto in precedenza.

Disporre di una zona di RAM non volatile è molto utile per il programmatore, poichè esiste la sicurezza che in essa rimarranno i dati più importanti ottenuti nel corso del programma e che si perderebbero dopo aver tolto l'alimentazione.

Gli altri registri sono identici a quelli del 6800.

Collegamenti alle periferiche nel 6802

Il microprocessore 6802 dispone di una PIA progettata su misura, con codice 6846, alla quale si possono collegare le periferiche secondo il tipico modo di accoppiamento mediante PIA. Il circuito 6846, oltre a disporre della possibilità di otto uscite in parallelo, possiede anche terminali per il bus di controllo e interrupts, ed un contatore-temporizzatore interno a cui si accede tramite programma e che serve per ottenere ritardi, conteggi per display, ecc.

Caratteristiche del µP 6502

Il microprocessore che costituisce la CPU è il 6502 della Rockwell.
Le sue caratteristiche sono:

- alimentazione a 5 V;
- costruito in tecnologia MOS, canale N;
- bus dei dati a otto bit;

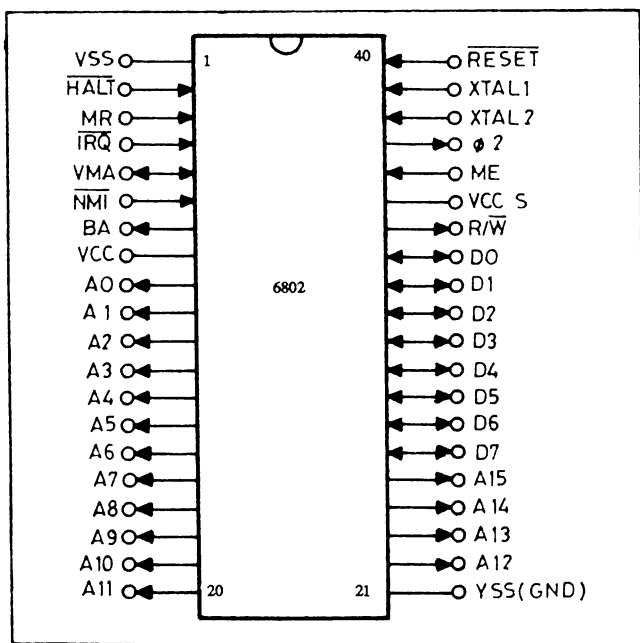



Fig. 8.-Terminali del microprocessore 6802 e loro funzioni.

- 56 istruzioni fondamentali;
- aritmetica in binario e decimale;
- modi di indirizzamento: diretto, relativo, immediato, indicizzato, implicito e indiretto;
- interrupts: logici (BRK), fisici (IRA), interrupt prioritario non mascherato (NMI);
- capacità di memoria: fino a 64 Kbytes;
- clock: da 1 MHz a 2 MHz.

In Fig. 9 si vede il contenitore a 40 terminali con il relativo pin-out che è il seguente:

- DB0-DB7: bus dei dati;
- AB0-AB15: bus degli indirizzi;
- R/W: segnale di lettura (R read) e scrittura (W write), per indicare alla memoria o alle periferiche quale sarà l'operazione successiva;

- RESET: azzeramento di tutti i registri interni per l'inizializzazione;
- RDY: segnale di sincronismo per le memorie lente;
- IRQ: richiesta di interrupt mascherabile;
- NMI: richiesta di interrupt non mascherabile;
- SYNC: segnale indicatore del ciclo di ricerca dell'istruzione;
- SO: flag di overflow;

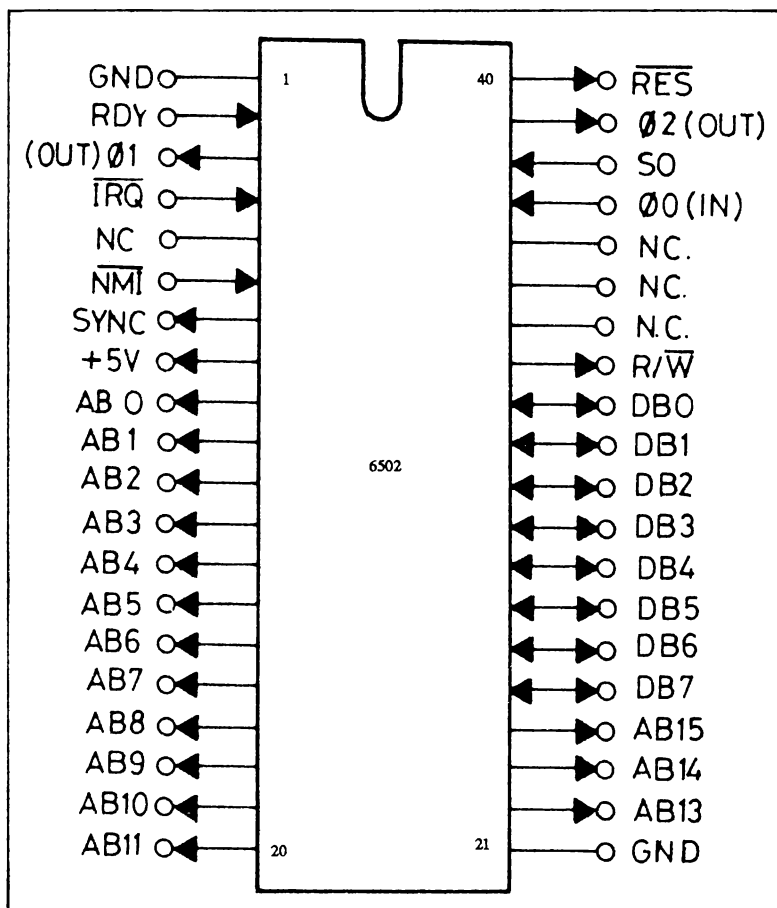



Fig. 9.-Aspetto esterno e distribuzione delle funzioni per ogni PIN del µP 6502

- GND: massa (-);
- +5: alimentazione a 5 V;
- Φ_0: segnale di clock della CPU;
- Φ_1, Φ_2: segnale di clock del sistema;
- NC: terminale non collegato.

Configurazione interna del µP

La sua configurazione interna è quella caratteristica di tutti i µP, ma la sua particolarità è di disporre di sei registri interni:

- 1 accumulatore a otto bit per uso generale, detto A;
- 2 puntatori a otto bit ciascuno, utilizzati nei registri indice X e Y;
- 1 contatore di programma a 16 bit (gli otto più alti PCH e gli otto più bassi PCL);
- 1 puntatore di stack per il rientro da subroutine a otto bit detto S;
- 1 registro di stato, i cui bit hanno il significato seguente:

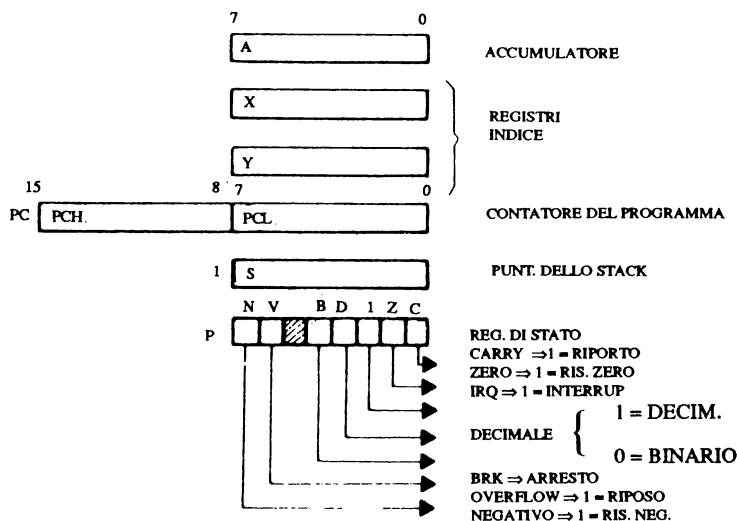



Fig. 10.-Registri interni del µP 6502.

0-C bit di carry (Carry);
1-Z bit di zero (Zero);
2-I abilitazione interrupt;
3-D bit di operazione binaria o decimale;
4-B break per l'arresto completo;
5 non usato
6-V bit di overflow;
7-N bit di segno.

E' utile esaminare più dettagliatamente i bit del registro di stato:

- 0-C=bit di carry: indica se l'operazione precedente ha provocato o no un riporto, se il riporto si è verificato viene settato a 1;
- 1-Z=bit di zero: indica se l'operazione precedente ha avuto risultato zero, se il bit è uno il risultato zero si è verificato, altrimenti il bit è a zero;
- 2-I=abilitazione interrupt: se il bit è 1 l'interrupt è permesso, se zero non permesso;
- 3-D=bit di operazione binaria o decimale: se vale uno il modo è decimale, se zero, binario;
- 4-B=break di arresto: se 0 non esiste arresto, se uno il µP si blocca completamente;
- 5=non utilizzato;

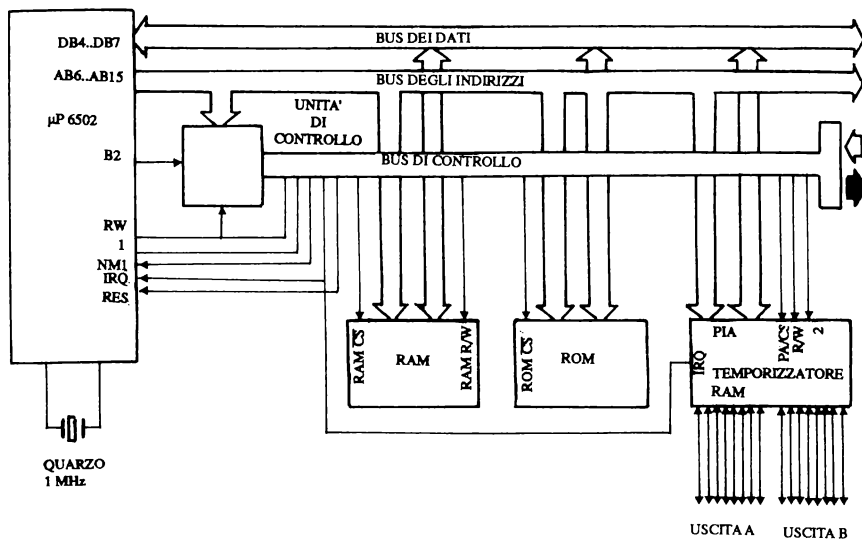



Fig. 11.-Blocchi funzionali del Junior Computer con dettaglio dei suoi bus.

- 6-V=bit di overflow: se viene settato a uno si è verificato un overflow;
- 7-N=bit di segno: se è uno indica che l'operazione precedente ha avuto risultato negativo.

Istruzioni del μP 6502

Il μP 6502 dispone di 56 istruzioni di programmazione, che sono riferite ai registri interni ed al modo di indirizzamento.

Le descrizioni mnemoniche sono in inglese, poichè il campo dell'informatica si è sviluppato nei paesi con tale lingua. Tali istruzioni sono:

- di caricamento (LOAD): LDA, LDX, LDY carica il contenuto della memoria nel registro specificato (A, X, Y);
- di memorizzazione (STORE): STA, STX, STY memorizza il contenuto del registro specificato nella memoria;
- di trasferimento (TRANSFER): TAX, TAY, ecc. trasferisce il dato tra i registri specificati;
- di somma (ADD): ADC somma la memoria con l'accumulatore;
- di differenza (SUBTRACT): SBC sottrae l'accumulatore e la memoria;
- logiche: AND, ORA, EOR esegue una operazione logica tra la memoria e l'accumulatore;
- di salto condizionato (BRANCH): BBC, BCS, ecc. salta da una locazione di memoria ad un'altra, se si verifica una condizione;
- di azzeramento e settaggio a uno dello status (CLEAR e SET): CLC, CLD, ecc., e SEC, SED, ecc. azzera e setta a uno, rispettivamente, il registro di stato indicato dall'istruzione;
- di confronto (COMPARE): CMP, CPX, ecc. confronta il registro indicato con la memoria;
- di incremento e decremento (INCREMENT, DECREMENT): INC, INX, ecc., e DEC, DEX, ecc. incrementa o decrementa di un'unità il registro indicato dall'istruzione;
- di salto (JUMP): JMP, JSR salta alla locazione di memoria indicata;
- di rotazione (ROTATE): ROL, ROR, ecc. ruota verso sinistra o destra il registro indicato dall'istruzione;

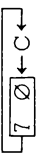
ADC	Add Memory to Accumulator with Carry
AND	«AND» Memory with Accumulator
ASL	Shift left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLd	Clear Decimal Mode
CLI	Clear Interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index X by One
EOR	«Exclusive» Memory with Accumulator
INC	Increment Memory by One
INC	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift One Bit Right (Memory or Accumulator)
NOP	No Operation
ORA	«OR» Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Register
TYA	Transfer Index Y to Accumulator

I \ J	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRK	ORA ind X				ORA PO	ASL PO		PHP	ORA imm	ASL A			ORA abs	ASL abs	
1	BPL	ORA ind Y				ORA POX	ASL POX		CLC	ORA abs Y				ORA absX	ASL absX	
2	JSR	AND ind X			BIT PO	AND PO	ROL PO		PLP	AND imm	ROL A		BIT abs	AND abs	ROL abs	
3	BMI	AND ind Y				AND POX	ROL POX		SEC	AND abs Y				AND absX	ROL absX	
4	RTI	EOR ind X				EOR PO	LSR PO		PHA	EOR imm	LSR A		JMP abs	EOR abs	LSR abs	
5	BVC	EOR ind Y				EOR POX	LSR POX		CLI	EOR abs Y				EOR absX	LSR absX	
6	RTS	ADC ind X				ADC PO	ROR PO		PLA	ADC imm	ROR A		JMP ind	ADC abs	ROR abs	
7	BVS	ADC ind Y				ADC POX	ROR POX		SEI	ADC abs Y				ADC absX	ROR absX	
8		STA ind X			STY PO	STA PO	STX PO		DEY		TXA		STY abs	STA abs	STX abs	
9	BCC	STA ind Y			STY POX	STA POX	STX POY		TYA	STA abs Y	TXS			STA absX		
A	LDY imm	LDA ind X	LDX imm		LDY PO	LDA PO	LDX PO		TAY	LDA imm	TAX		LDY abs	LDA abs	LDX abs	
B	BCS	LDA ind Y			LDY POX	LDA POX	LDX POY		CLV	LDA abs Y	TSX		LDY absX	LDA absX	LDX absY	
C	CPY imm	CMP ind X			CPY PO	CMP PO	DEC PO		INY	CMP imm	DEX		CPY abs	CMP abs	DEC abs	
D	BCS	CMP ind Y				CMP POX	DEC POX		CLD	CMP abs Y				CMP absX	DEC absX	
E	CPX imm	SBC ind X			CPX PO	SBC PO	INC PO		INX	SBC imm	NOP		CPX abs	SBC abs	INC abs	
F	BEQ	SBC ind Y				SBC POX	INC POX		SED	SBC abs Y				SBC absX	INC absX	

Tabella III.-Mappa delle istruzioni del μP 6502. Esempio di utilizzo: 94=STY, pagina 0 indicizzato tramite X.

TABELLA IV

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di impulsi di clock*	*Numero di bytes*	*Flag influenzati*
ADC Add memory to accumulator with carry A+M+C → A(1) **Sommare M e A con carry**	IMM ABS Z (IND, X) (IND), Y ZX ABS, X ABS, Y	69 6D 65 61 71 75 7D 79	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	NV..........ZC
AND «AND» memory with accumulator A ∧ M → A(1) **Funzione AND tra M e A**	IMM ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	29 2D 25 21 31 35 3D 39	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	N..........Z..
ASL Shift left one bit **Spostare un bit a sinistra** C ← [7 Ø] ← Ø	ABS Z A Z, X ABS, X	ØE Ø6 ØA 16 1E	6 5 2 6 7	3 2 1 2 3	N..........ZC
BCC Branch on carry clear (2) **Salto se C=0**	REL	9Ø	2	2	
BCS Branch on carry set (2) **Salto se C=1**	REL	BØ	2	2	

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di impulsi di clock*	*Numero di bytes*	*Flag influenzati*
BEQ Branch on result zero (2) **Salto se Z=1**	REL	FØ	2	2	
BIT Test bits in memory **Eseguire AM** $M_7 \rightarrow N$; $M_8 \rightarrow V$	ABS Z	2C 24	4 3	3 2	M_7 M_8 ....Z
BMI Branch on result minus (2) **Salto se N=1**	REL	3Ø	2	2	
BNE Branch on result not zero (2) **Salto se Z=0**	REL	DØ	2	2	
BPL Branch on result plus (2) **Salto se N=0**	REL	1Ø	2	2	
BRK Force break **Interruzione obbligata**	IMP	ØØ	7	1	..............................1...1... B 1
BVC Branch on overflow clear (2) **Salto se V=0**	REL	5Ø	2	2	

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di impulsi di clock*	*Numero di bytes*	*Flag influenzati*
BVS Branch on overflow set (2) **Salto se V=1**	REL	7Ø	2	2	
CLC Clear carry flag **Disattivare C; 0-C**	IMP	18	2	1	..........Ø C
CLD Clear decimal mode Desactivar **Disattivare D; 0-D**	IMP	D8	2	1	.....Ø..... D
CLI Clear interrupt flag **Disattivare I; 0-I**	IMP	58	2	1	..........Ø.. i
CLV Clear overflow flag **Disattivare V; 0-V**	IMP	B8	2	1	Ø.......... V
CMP Compare memory and accumulator A-M **Confrontare M e A**	IMM ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	C9 CD C5 C1 D1 D5 DD D9	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	N..........ZC

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di impulsi di clock*	*Numero di bytes*	*Flag influenzati*
CPX Compare memory and index X; **Confrontare M e Y**	IMM ABS Z	EØ EC E4	2 4 3	2 3 2	N..........ZC
CPY Compare memory and index Y **Confrontare M e Y; M-Y**	IMM ABS Z	CØ CC C4	2 4 3	2 3 2	N..........ZC
DEC Decrement memory **Togliere 1 da M** **M-1 → M**	ABS Z Z, X ABS, X	CE C6 D6 DE	6 5 6 7	3 2 2 3	N..........Z..
DEX Decrement index X by one **Togliere 1 da X**	IMP	CA	2	1	N..........Z..
DEY Decrement index Y by one Y-1 → Y **Togliere 1 da Y**	IMP	88	2	1	N..........Z..
EOR «Exclusive or» memory with accumulator A ∀ M → A (1) **Funzione XOR tra M e A**	IMM ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	49 4D 45 41 51 55 5D 59	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	N..........Z..

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di impulsi di clock*	*Numero di bytes*	*Flag influenzati*
INC Increment memory by one X+1 → M **Sommare 1 a M**	ABS Z Z, X ABS, X	EE E6 F6 FE	6 5 6 7	3 2 2 3	N..........Z...
INX Increment index X by one X+1 → X **Sommare 1 a X**	IMP	E8	2	1	N..........Z...
INY Increment index Y by one Y+1 → Y **Sommare 1 a Y**	IMP	C8	2	1	N..........Z...
JMP Jumpto new loc. **Salto** (PC+1) → PCL (PC+2) → PCH	ABS IND	4C 6C	3 5	3 3	
JSR Jumpto to new location saving retunr address **Salto con ritorno** (PC+1) → PCL (PC+2) → PCH	ABS	2Ø	6	3	

Abbreviazione mnemonica+ spiegazione	Modo di indirizzamento (5)	Codice esadecimale	Numero di impulsi di clock	Numero di bytes	Flag influenzati
LDA Load accumulator with memory M → A (1) **Caricare A con M**	IMM ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	A9 AD A5 A1 B1 B5 BD B9	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	N............ ....Z...
LDX Load index X with memory M → X (1) **Caricare X con M**	IMM ABS Z Z, Y ABS, Y	A2 AE A6 B6 BE	2 4 3 4 4	2 3 2 2 3	N..............................Z...
LDY Load Index Y with memory M → Y (1) **Caricare Y con M**	IMM ABS Z Z, X ABS, X	AØ AC A4 B4 BC	2 4 3 4 4	2 3 2 2 3	N..............................Z...
LSR Shift right one bit **Spostare un bit a sinistra** Ø → [7 Ø] → C	ABS Z A Z, X ABS, X	4E 46 4A 56 5E	6 5 2 6 7	3 2 1 2 3	Ø.............................ZC N
NOP No operation	IMP	EA	2	1	

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di* **impulsi di clock**	**Numero di bytes**	**Flag influenzati**
ORA «OR» memory with accumulator AVM → A **Funzione OR tra M e A**	IMM ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	Ø9 ØD Ø5 Ø1 11 15 1D 19	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	N..............................Z...
PHA Push accumulator on stack AI **Inviare A allo stack**	IMP	48	3	1	
PHP Push processor status on stack; PI **Inviare P allo stack**	IMP	Ø8	3	1	
PLA Pull accumulator from stack AI **Ricuperare A dallo stack**	IMP	68	4	1	N..............................Z...
PLP Pull proceşor status from stack: PI **Recuperare P dallo stack**	IMP	28	4	1	(reset)

Abbreviazione mnemonica+ spiegazione	Modo di indirizzamento (5)	Codice esadecimale	Numero di impulsi di clock	Numero di bytes	Flag influenzati
ROL Rotate one bit left **Ruotare un bit a sinistra** └[7 Ø]←C←┘	ABS Z Z, X ABS, X A	2E 26 36 3E 2A	6 5 6 7 2	3 2 2 3 1	N..........ZC
ROR Rota, one bit right **Ruotare un bit a destra** └→C→[7 Ø]┘	ABS Z A Z, X ABX, X	6E 66 6A 76 7E	6 5 2 6 7	3 2 1 2 3	N..........ZC
RTI Return from interrupt PCI, PI **Ritorno da interrupt** rup.	IMP	4Ø	6	1	(reset)
RTS Return from subroutine PCI; PC+1-PC **Ritorno da subroutine**	IMP	6Ø	6	1	
SBC Subtract memory from accumulator with borrow (3) $A\text{-}M\text{-}\overline{C} \rightarrow A$ **Sottrarre M e carry negato da A**	IMM ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	E9 ED E5 E1 F1 F5 FD F9	2 4 3 6 5 4 4 4	2 3 2 2 2 2 3 3	N..........ZC

Abbreviazione mnemonica+ spiegazione	*Modo di indirizzamento (5)*	*Codice esadecimale*	*Numero di impulsi di clock*	*Numero di bytes*	*Flag influenzati*
SEC Set carry flag 1 → C **Settare C**	IMP	38	2	1	1
SED **Settare D** var D	IMP	F8	2	1	................1......... D
SEI Setinterrupt; 1 → I **Settare I**	IMP	78	2	1	......................1... I
STA Store accumulator in memory A → M **Memorizzare A in M**	ABS Z (IND, X) (IND), Y Z, X ABS, X ABS, Y	8D 85 81 91 95 9D 99	4 3 6 6 4 5 5	3 2 2 2 2 3 3	
STX (X → M) Storex index X in M **Memorizzare Y in M** M	ABS Z Z, Y	8E 86 96	4 3 4	3 2 2	
STY (Y → M) Store index Y in M **Memorizzare X in M**	ABS Z Z, X	8C 84 94	4 3 4	3 2 2	

Abbreviazione mnemonica+ spiegazione	Modo di indirizzamento (S)	Codice esadecimale	Numero di impulsi di clock	Numero di bytes	Flag influenzati
TAX (A → X) Transfer accumulator to index X **Trasferire A a X**	IMP	AA	2	1	N.......Z...
TAY (A → Y) Transfer accumulator to index Y **Trasferire A a Y**	IMP	A8	2	1	N.......Z...
TSX (S → X) Transfer stack pointer to index X **Trasferire S a X**	IMP	BA	2	1	N.......Z...
TXA (X → A) Transfer index X to accumulator **Trasferire X a A**	IMP	8A	2	1	N.......Z...
TXS (X → S) Transfer index X to stack pointer **Trasferire X a S**	IMP	9A	2	1	N.......Z...
TYA (Y → A) Transfer index Y to accumulator **Trasferire Y a A**	IMP	98	2	1	N.......Z...

Tabella IV.-Elenco delle istruzioni del μP 6502.

- di copia (PUSH, PULL): PHA, PHP, ecc. trasferisce il dato tra il registro indicato e lo stack;
- di ritorno (RETURN): RTI, RTS ritorna da subroutine;
- di non operare: NOP questa istruzione non esegue alcunchè, e agisce da riempitivo.

Note relative alla tabella delle istruzioni

(1) Sommare 1 a N se si supera il limite di pagina. (2) Sommare 1 a N se il salto si esegue alla stessa pagina, sommare 2 a N se il salto si esegue ad altra pagina. (3) Senza carry. (4) A=accumulatore, M=memoria, C=flag di carry, Z=flag di segno, V=flag di overflow, N=flag di segno negativo, D=flag di numero decimale, I=flag di interrupt, X=indice X, Y=indice Y.

(5)

IMM:	indirizzamento immediato;
ABS:	indirizzamento assoluto;
Z:	indirizzamento in pagina zero;
A:	indirizzamento dell'accumulatore;
IMP:	indirizzamento implicito;
(IND, X)	indirizzamento preindicizzato;
(IND, Y)	indirizzamento preindicizzato;
Z, X	indirizzamento indicizzato in pagina zero (utilizza registro);
Z, Y	indirizzamento indicizzato in pagina zero (utilizza registro);
ABS, X	indirizzamento assoluto indicizzato (utilizza registro);
ABS, Y	indirizzamento assoluto indicizzato (utilizza registro);
REL	indirizzamento relativo;
IND	indirizzamento diretto.

Modi di indirizzamento del 6502

Le varie istruzioni di questo μP dispongono dei seguenti modi di indirizzamento:

- Indirizzamento dell'accumulatore:

  Sono istruzioni a un solo byte che si riferiscono ad una operazione con il registro accumulatore (A).
- Indirizzamento immediato:

  Sono istruzioni a due bytes, di cui il primo è l'istruzione e il secondo l'operando; questo indirizzamento non necessita di

indirizzo di memoria.

- Indirizzamento assoluto:

  Le istruzioni che utilizzano questo indirizzamento sono a tre bytes: il primo è l'istruzione, il secondo e terzo costituiscono l'indirizzo di memoria in cui si vuole operare. Con questo tipo di indirizzamento è possibile coprire i 64 Kbytes di cui dispone il µP 6502.

- Indirizzamento in pagina zero:

  Utilizzano questo tipo di indirizzamento le istruzioni che operano solo in pagina zero.

- Indirizzamento indicizzato in pagina zero:

  Le istruzioni che utilizzano questo indirizzamento, lo fanno lavorando in pagina zero con l'indicizzazione del registro X e Y.

- Indirizzamento indicizzato assoluto:

  Questo tipo di indirizzamento è come quello assoluto, ma indicizzato con i registri X e Y.

  L'indirizzo effettivo è il risultato della somma del contenuto del registro X o Y, e dell'indirizzo assoluto indicato dal secondo e terzo byte.

- Indirizzamento implicito:

  Nel modo implicito, l'indirizzo dell'operazione è contenuto nell'operando.

- Indirizzamento relativo:

  Questo tipo di indirizzamento è utilizzato solo con le istruzioni di salto. Sono istruzioni a due bytes, in cui il secondo indica se il salto è in avanti o all'indietro, ed il numero di locazioni da saltare.

  I limiti del salto sono di 127 posizioni in avanti e 128 all'indietro rispetto alla posizione in cui si trova l'istruzione che si sta eseguendo.

- Indirizzamento indicizzato indiretto:

  L'indirizzo finale dell'operazione si trova nella locazione di memoria indicata mediante l'indirizzamento indicizzato; il registro indice può essere X o Y.

- Indirizzamento assoluto indiretto:

  L'indirizzo effettivo si trova mediante un indirizzamento assoluto.

CAPITOLO 7

MICROPROCESSORI A 16 BIT

'espansione dei campi di applicazione e delle possibilità di impiego dei microprocessori a 8 bit, è stata così ampia che, a partire dal 1980, questi microprocessori erano utilizzati al limite delle loro possibilità. Inoltre erano prevedibili applicazioni che superavano tali possibilità.

Per soddisfare queste necessità di mercato nacquero i microprocessori a 16 bit. Quelli a 32 bit furono realizzati subito dopo.

Quanto accadde ai microprocessori a 4 bit con l'avvento di quelli a 8 bit, sta ora accadendo a questi ultimi con l'avvento di quelli a 16.

D'altra parte, chi è in grado di sviluppare sistemi con i microprocessori a 8 bit, non avrà molti problemi con quelli a 16 bit, vista la loro somiglianza. Analogamente accadrà con quelli a 32 bit.

L'obiettivo di questo capitolo è di descrivere, in modo semplice, la filosofia dei microprocessori a 16 bit a un lettore che conosca quelli a 8 bit.

Perciò vengono forniti solo i concetti basilari, relativi all'hardware e al software, per permettere di *iniziare a lavorare* con i nuovi componenti. Infatti solo l'effettiva sperimentazione aiuta, quando si inizia a trattare in modo approfondito gli argomenti chiave.

Tra i microprocessori a 16 bit, sono stati scelti solo quelli per cui si possono prevedere più applicazioni:

- 1. Il classico 8086, che è il più venduto nella sua categoria, e che è supportato da una eccellente famiglia di circuiti ausiliari. L'inserimento dell'8088, versione dell'8086 con il bus dei dati a 8 bit, nel personal computer IBM, fa prevedere un grande sviluppo dell'hardware e del software di tale famiglia.
- 2. Il 68000, che possiede un'architettura protesa verso i 32 bit, e

un impianto software orientato ai sistemi operativi moderni ed ai linguaggi strutturali ad alto livello.

Generazione dei microprocessori a 16 bit

La scoperta della tecnologia HMOS, da parte dell'INTEL CORPORATION, ha propiziato l'aumento della densità di integrazione, e il passaggio alla tecnologia VLSI, con cui sono stati ampiamente superati i 100.000 transistori in un chip, fa sperare di raggiungere tra breve il milione.

Storicamente INTEL fu la prima costruttrice di un microprocessore a 16 bit, all'inizio del 1978. Dopo l'8086 dell'INTEL, apparvero lo Z8000 della ZILOG, il 68000 della MOTOROLA, l'NS 1600 della NATIONAL, il 99000 della TEXAS, il µCOM 70K, successore del 6502 della NEC, e altri ancora.

In questo momento l'applicazione più importante dei microprocessori a 16 bit è quella nei microelaboratori ad uso generale.

La nascita dei microprocessori a 16 e 32 bit è stata necessaria per adeguarli ai requisiti delle nuove applicazioni, ed al miglioramento e semplificazione del software.

L'hardware di questi microprocessori, seguendo in molti casi la linea degli 8 bit, si orienta verso la simmetria e la possibilità di ampliamenti e variazioni, come dimostra l'inserimento in molti modelli dell'Unità di Controllo in Multiprogrammazione.

Parallelamente, l'uso delle nuove tecnologie, come l'HMOS, nella fabbricazione dei circuiti integrati, permette di aumentare la densità di integrazione e migliorare il rapporto velocità/consumo.

Tuttavia, il miglioramento più importante dei moderni microprocessori è relativo al software. Il suo costo ed i problemi che presenta all'utente hanno spinto i progettisti ad una architettura speciale, che semplifichi e minimizzi tutti questi fattori.

Così, nella maggioranza dei microprocessori di progettazione recente, esistono caratteristiche sconosciute in quelli a 8 bit, come:

- 1. repertorio delle istruzioni orientato verso l'implementazione di linguaggi ad alto livello;
- 2. elevata varietà dei modi di indirizzamento;
- 3. grande diversità nei formati dei dati;
- 4. possibilità di lavorare in memoria virtuale, mediante la quale il microprocessore è in grado di operare in un campo di memoria più ampio di quello che esiste effettivamente nel sistema;
- 5. impiego del modo Supervisor, per la protezione dei programmi,

MODELLO	FABBRICANTE	FORMATO DEI DATI	MULTIPLEXER DEI DATI	REGISTRI	FREQUENZA DI CLOCK	CAPACITA' MEMORIA	ISTR. PIU' LUNGA	CODA DI ATTESA DELLE ISTRUZIONI	SECONDA FONTE
8086	INTEL	16/16	SI	14 A 16 BITS DI USO NON GENERALE	4, 5 y 8 MHz	7 M BYTES	20 μ s	6 BYTES	AMD, FUJITSU HARRIS, MATRA MITSUBISHI, NEC E SIEMENS
8001	ZILOG	16/16	SI	7 + 4 A 16 BITS SPECIALIZZATI	4, 6 Y 10 MHz	8 M BYTES	140 μ s	NO	AMD, SOS E SHARP
68000	MOTOROLA	32/16	NO	18 A 32 BITS + 1 A 16 BITS SPECIALIZZATI	4, 6, 8, 10 y 12 MHz	16 M BYTES	20 μ s	NO	HITACHI, MOSTEK, ROCKWELL, SIGNETICS E THOMPSON
99000	TEXAS	16/16	NO	8 A 16 BITS SPECIALI	4 MHz	64 K BYTES	31 μ s	NO	AMI, ITT
16032	NATIONAL	32/16	SI	6 A 24 BITS + 2 A 16 BITS SPECIALI E 8 A 32 BITS GENERALI	10 MHz	16 M BYTES	8 μ s	8 BYTES	EUROTECNIQUE E FAIRCHILD

Tabella I.-Riepilogo delle caratteristiche più importanti di alcuni microprocessori a 16 bits.

e adeguamento ai moderni sistemi operativi;

- 6. sistema completo di interrupts;
- 7. nuove istruzioni, inesistenti in quelli a 8 bit, come moltiplicazione, divisione, trasferimento blocchi di dati, ecc.

Come già accadde con i 4 bit, i microprocessori a 8 bit verranno soppiantati dai 16 e 32 bit, il che non significa che saranno completamente accantonati.

D'altra parte il piccolo aumento di prezzo dei microprocessori a 16 e 32 bit, rispetto alle loro straordinarie possibilità, contribuirà a renderli presto di uso comune.

Il microprocessore 68000

L'aspetto più rilevante del 68000 sta nel fatto che può essere considerato come un effettivo microprocessore a 32 bit, anche se viene compreso tra quelli a 16. La versione siglata 68020 è disponibile in commercio, come CPU a 32 bit, dal 1983.

Nel 68000 sono riunite una serie di caratteristiche che lo evidenziano tra i microprocessori, come l'assoluta flessibilità per la realizzazione di sistemi multiprocessore o multiarea, unita ad altre che gli permettono di adattarsi a prodotti standard, tipici dei microprocessori a 8 bit, memorie e moduli di ingresso e uscita, con formati e velocità di lavoro variabili.

Questa possibilità di utilizzare nella configurazione di sistemi basati sul 68000, elementi molto noti presenti in commercio, riduce significativamente i costi del progetto.

Il 68000 è simile ad un microprocessore a 32 bit suddivisi in due parole da 16 bit. Ciò è conseguenza del fatto che, non multiplexando i bus dei dati e degli indirizzi, sarebbe necessario un contenitore a 96 piedini per utilizzare tutti i segnali inerenti al formato a 32 bit.

Il costo e le restrizioni tecnologiche ne motivarono la produzione in contenitori a 64 terminali, ma il passaggio ai 32 bit è semplice ed è già stato effettuato.

Il 68000 ha migliorato molti aspetti del suo predecessore 6800:

- 1. è stato ampliato moltissimo il numero delle istruzioni, comprese quelle proprie dei 16 bit, come moltiplicazione e divisione. Altre, come PEA e LEA, consentono la programmazione strutturata;
- 2. il numero dei modi di indirizzamento è stato portato a 14, tra cui alcuni di indubbia operatività;

- 3. si è cercato di semplificare l'adattamento dei programmi del 6800 al 68000;
- 4. si è permessa l'elaborazione di dati con formato variabile da 1 a 32 bit;
- 5. sono stati incorporati segnali propri del 6800, come VPA, VMA, ed E, che lo rendono direttamente compatibile con i moduli di ingresso e uscita della famiglia del microprocessore a 8 bit.

I costruttori principali sono quelli sottoelencati:

- 1. MOTOROLA (MC6800);
- 2. ROCKWELL (R68000);
- 3. HITACHI AMERICA LTD. (HD68000);
- 4. SIGNETICS/PHILIPS (SP68000)
- 5. MOSTECK CORP. (MK68000);
- 6. EFCIS (THOMSON-CFS) (EF68000).

Architettura del 68000

Due grandi blocchi di registri a 32 bit, uno per i dati e l'altro per gli indirizzi, costituiscono la base su cui poggia l'hardware e, di conseguenza, il software di questo microprocessore.

Esistono inoltre un'Unità Aritmetico-Logica, un registro di stato, e il registro delle istruzioni, assieme alla logica di controllo e temporizzazione, come si vede in Fig. 1.

La ALU è in grado di effettuare operazioni logiche e aritmetiche con dati a 32 bit, comprese la moltiplicazione e la divisione, con e senza segno.

I rimanenti componenti del 68000 intervengono in modo molto diverso nelle istruzioni, per cui queste devono essere ben conosciute dal progettista.

I registri che intervengono, in modo preponderante, nella programmazione del 68000, così come il complesso di segnali che lo mettono in comunicazione con l'esterno, sono graficamente rappresentati in Fig. 2.

Registro dei dati

E' un banco di 8 registri a 32 bit ciascuno, destinati a contenere dati e operandi. Per dare un'idea della sua flessibilità, si può dire che equivale ad 8 accumulatori, di quelli visti nei microprocessori a 8 bit.

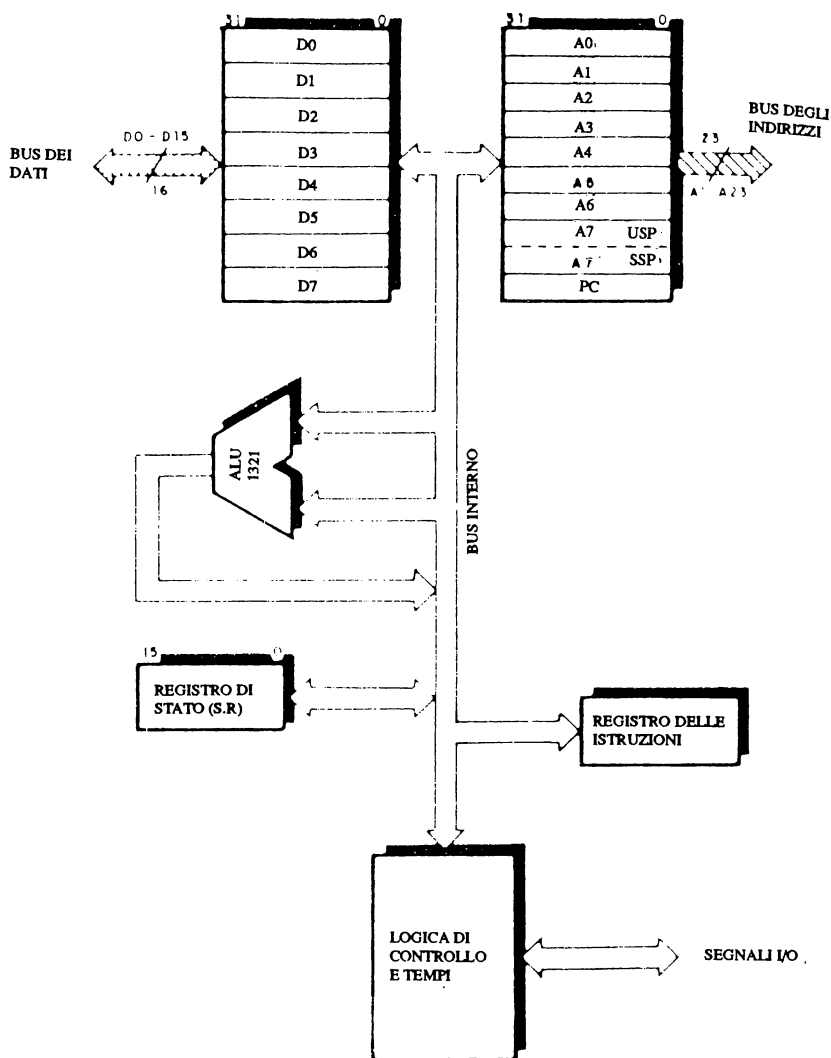



Fig. 1.-Architettura del 68000. La sua semplicità è unita ad una enorme flessibilità dei due banchi di registri, il che spiega la sua potenza operativa.

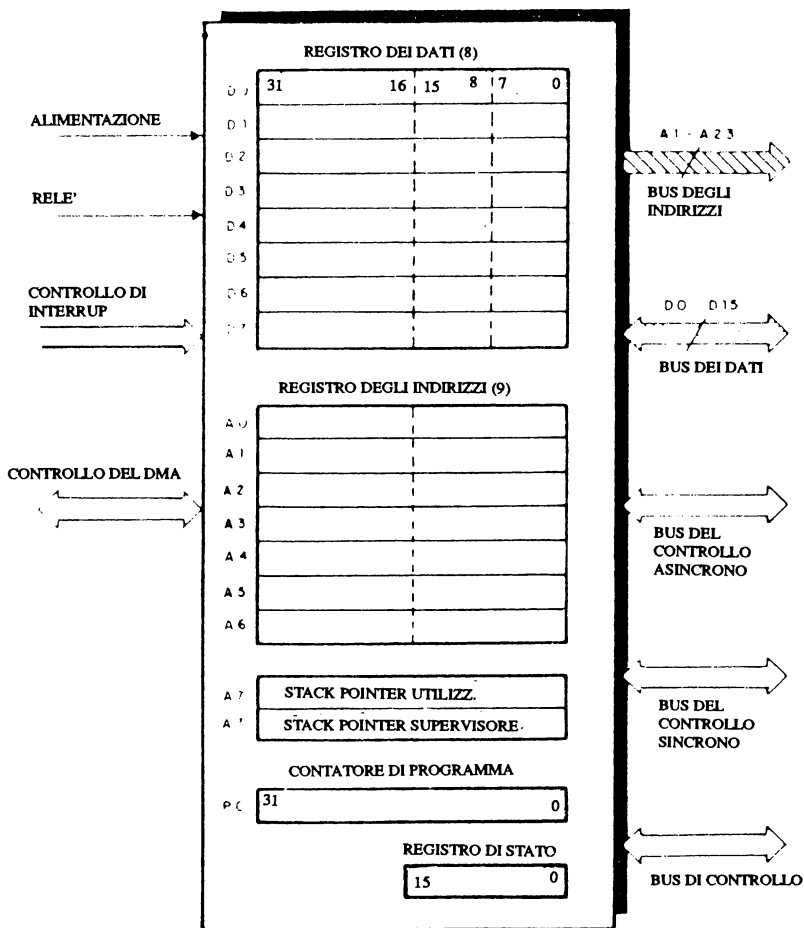



Fig. 2.-Organizzazione interna del 68000, dal punto di vista della programmazione dei banchi di registri.

L'informazione elaborata nei registri dei dati può essere a 8, 16, 32 bit.

Per determinare le dimensioni dell'informazione utilizzata in un registro di dati, si pone nel codice mnemonico della corrispondente istruzione, un punto seguito da B, W, o L, secondo che si tratti di 8, 16, o 32 bit, rispettivamente.

Le operazioni con i registri di dati influenzano solo i bit utilizzati nell'i-

struzione, gli altri non vengono variati, anche in istruzioni di spostamento.

Registri degli indirizzi

E' costituito da 7 registri a 32 bit, che hanno il compito di memorizzare gli indirizzi di memoria. Pertanto il loro contenuto deve essere correlato con il modo di indirizzamento delle diverse lunghezze dei dati.

Si osservi che gli indirizzi di memoria che si riferiscono a parole lunghe, si succedono di 4 in 4. Non rispettare queste norme provoca un *indirizzo illegale*.

Le dimensioni che possono essere utilizzate da registri degli indirizzi, quando agiscono su un operando sorgente dell'istruzione, sono di 16 e 32 bit.

Tuttavia, quando gli operandi sono di destinazione, tutti i 32 bit del registro sono influenzati, indipendentemente dalle dimensioni dell'operando.

Cioè se si elabora un operando a 16 bit, gli altri 16 del registro degli indirizzi destinazione sono influenzati, con l'estensione a tali bit del segno (il bit più significativo dei 16 bit del risultato viene ripetuto nei 16 bit non utilizzati).

Un altro importante dettaglio relativo a tali registri, consiste nel fatto che le operazioni che si effettuano con essi non influiscono sui flags di condizione (CCR) del registro di stato.

I registri puntatori di stack

I registri degli indirizzi A7 e A7' sono due puntatori di stack, che è una zona di memoria in cui si salva lo stato della CPU quando esiste un salto a subroutine o interviene un interrupt. Il registro A7 corrisponde al puntatore di stack dell'ambiente utente, l'A7' a quello del supervisore. In ogni istante solo uno dei due è in funzione.

Normalmente il puntatore di stack dell'ambiente utente funziona nelle chiamate a subroutine di tale ambiente. L'A7' funziona durante gli interrupts.

I due puntatori di stack indicano l'ultimo dato contenuto. Si caricano verso l'alto e si scaricano verso il basso.

Contatore di programma

E' un registro a 32 bit, che viene utilizzato per contenere l'indirizzo della parola dell'istruzione successiva a cui si deve accedere.

Sono utilizzati solo 24 bit, anche se il bus degli indirizzi viene caricato con soli 23, poichè non accoglie il bit meno significativo A0.

Con 23 bit di possono indirizzare 2^{23} = 8.388.608 parole a 16 bit.

Il bit meno significato del PC si utilizza per determinare all'interno dell'indirizzo, della parola cui si accede, se si riferisce al byte pari (LDS) o dispari (UDS).

Tenendo conto della funzione di questo bit, si possono indirizzare 16 Mbytes.

Se A0=0 l'indirizzo è pari e può essere riferito a un byte pari o a una parola. In questo caso, si attiva il segnale UDS, che esce dalla CPU, abilitando gli 8 bit più significativi del bus dei dati (D8-D15).

Se A0=1, l'indirizzo è dispari e si riferisce a un byte dispari. Si attiva il segnale LDS, che abilita gli 8 bit meno significativi del bus dei dati (D0-D7).

Registro di stato

E' costituito da 16 bit, suddivisi in due bytes, uno destinato al sistema e l'altro all'utente.

La sua struttura è quella di Fig. 3.

I cinque bit rappresentativi nel byte dell'utente (gli altri tre sono sempre 0) sono i flags o codici di condizione dell'operazione effettuata nell'ultima istruzione eseguita dalla CPU.

Il significato di ognuno di essi è il seguente:

- C: carry, viene settato a 1 quando esiste riporto nei bit più significativi di una somma, o prestito per una differenza. Interviene in altre istruzioni, come quelle di spostamento, rotazione, ecc.;
- V: overflow, ha significato solo nelle operazioni in cui intervengono numeri con segno. Diventa 1 quando si sommano due numeri con lo stesso segno o se ne sottraggono due di segno opposto ed il risultato supera il campo dell'operando in complemento a 2;
- Z: zero, questo bit viene settato a 1 quando il risultato dell'operazione vale zero;
- N: segno, ha significato solo quando si lavora con numeri con segno. Diventa 1 quando, dopo un'operazione, il risultato è negativo, cioè il bit più significativo è 1.
- X: carry esteso, è un bit di carry (C), per operazioni in multipla precisione. Serve per concatenare operazioni in cui intervengono

BYTE DEL SISTEMA								BYTE DELL'UTENTE							
T	0	S	0	0	I_2	I_1	I_0	0	0	0	X	N	Z	V	C

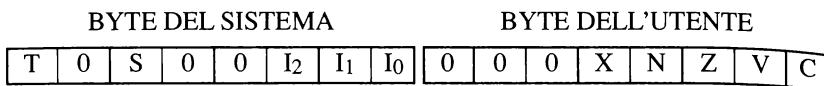



Fig. 3.-Configurazione e denominazione dei 16 bits del registro di stato (SR).

dati di lunghezza maggiore. E' trasparente al flusso dei dati e, nelle istruzioni in cui agisce, ha lo stesso valore del registro C.

Anche il byte del registro di stato relativo al sistema possiede 5 bit significativi, con i seguenti significati:

- T: modo trace, si può settare a 1 solo tramite lo stato supervisore, il che può accadere dopo ogni istruzione, per eseguire un programma passo a passo, e permettere all'utente di conoscere lo stato del sistema. Per ogni eccezione esiste un byte che rappresenta un vettore il quale, moltiplicato per 4, fornisce l'indirizzo che contiene la locazione di inizio del programma di gestione dell'eccezione. Per implementare il trace, istruzione per istruzione, si accede a un vettore che serve una routine in grado di esaminare i registri della CPU, le locazioni di memoria, ecc;
- S: Supervisore, indica quando il 68000 funziona in modo supervisore (S=1). In questo modo, tutte le istruzioni sono eseguibili, e si può modificare la maschera degli interrupts (I_0-I_2). Si può anche accedere al registro di stato e ai puntatori di stack;

LIVELLO	*I_0*	*I_1*	*I_2*	
0	0	0	0	SENZA PRIORITA'
1	1	0	0	MINIMA PRIORITA'
2	0	1	0	
3	1	1	0	
4	0	0	1	
5	1	0	1	
6	0	1	1	
7	1	1	1	MASSIMA PRIORITA' (NON MASCHERABILE)

Tabella II.-Tabella dei livelli di priorità che riflette la maschera degli interrupts, costituita dai bits I_0-I_2. Tutte le richieste di interrupt con un livello uguale o inferiore a quello stabilito con la maschera, non sono accettati dalla CPU.

- (I_0-I_2): maschera degli interrupts, la loro combinazione logica riflette il livello di priorità a partire dal quale si accettano gli interrupts. Nella tabella II sono visibili i vari livelli di priorità.

Configurazione di un sistema basato sul 68000

I 64 terminali del 68000 implementano tutte le funzioni necessarie per sviluppare un potente complesso di bus di dati, indirizzi, e controllo, che permettono di costruire un ottimo sistema microelaboratore, il cui schema di principio è visibile in Fig. 4.

A tale sistema si possono accoppiare tutti i tipi di memoria e moduli di ingresso e uscita standard, come quelli della famiglia del 6800.

L'esistenza di due bus di controllo, uno sincrono e l'altro asincrono, amplia notevolmente la possibilità di inserire una gran varietà di periferiche nel sistema.

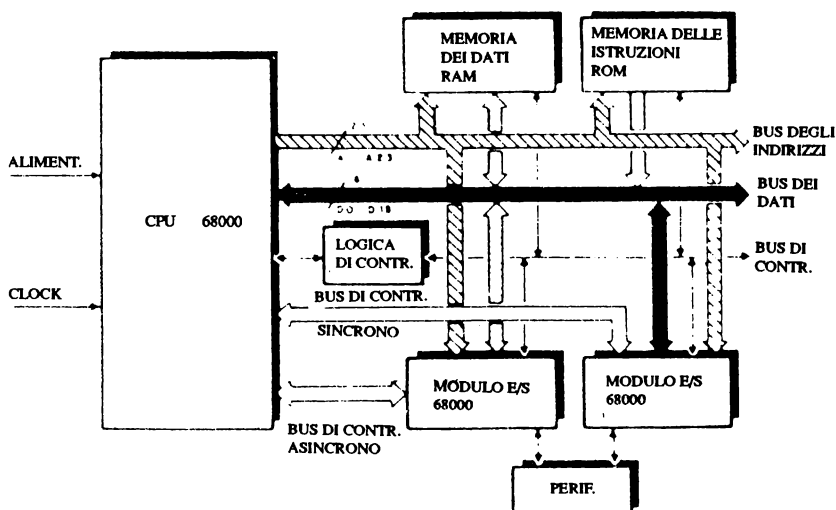



Fig. 4.-Configurazione fondamentale di un sistema microelaboratore, basato sul 68000.

Schema generale di collegamento del 68000

Il chip del 68000 è inserito in un contenitore di tipo DIL a 64 piedini.

Oltre al bus dei dati a 16 linee (D0-D15) e al bus degli indirizzi a 23 (A1-A23), esistono altri segnali che si possono raggruppare nel modo seguente:

- 1. bus di controllo sincrono per periferiche della famiglia 6800;
- 2. bus di controllo asincrono per periferiche del 68000;
- 3. controllo di DMA o di gestione del bus;
- 4. controllo della funzione o stato della CPU;
- 5. controllo del sistema.

La Fig. 5 indica la numerazione dei terminali assegnati ad ogni linea. I segnali rappresentati in modo sopralineato sono attivi quando il loro livello logico è basso. Per una miglior comprensione si descrive il compito delle linee del microprocessore raggruppate per funzioni:

- alimentazione: esistono due piedini, il 14 e il 49, mediante i quali si fornisce la tensione di alimentazione positiva il cui valore deve essere di 5 V, con una tolleranza del 5%. I terminali 16 e 53 si collegano al polo negativo o massa (GND).
- clock: l'ingresso corrispondente al piedino 15 del circuito integrato, fornisce un segnale di frequenza stabile, TTL compatibile, che serve per ottenere internamente gli impulsi del clock di sincronismo.
- bus dei dati: è costituito da 16 linee bidirezionali, denominate D0-D15, che sono distribuite, strategicamente, sui terminali della zona superiore del contenitore, allo scopo di semplificare il disegno delle piste che convogliano l'informazione sul circuito stampato. La limitazione del numero delle linee del bus dei dati implica che per eseguire il trasferimento con un numero di bit superiore alla parola, è necessario più di un ciclo.
- bus degli indirizzi e segnali UDS e LDS: è costituito da 23 linee unidirezionali, che escono dalla CPU, mediante i terminali detti A1-A23. I segnali UDS e LDS selezionano il byte pari o dispari di una parola, per cui, considerando anch'essi, si possono indirizzare fino a 16 Mbytes.
- bus di controllo asincrono: la funzione delle linee principali che costituiscono il bus di controllo asincrono del 68000, per il controllo di periferiche caratteristiche della famiglia 68000, è la seguente:

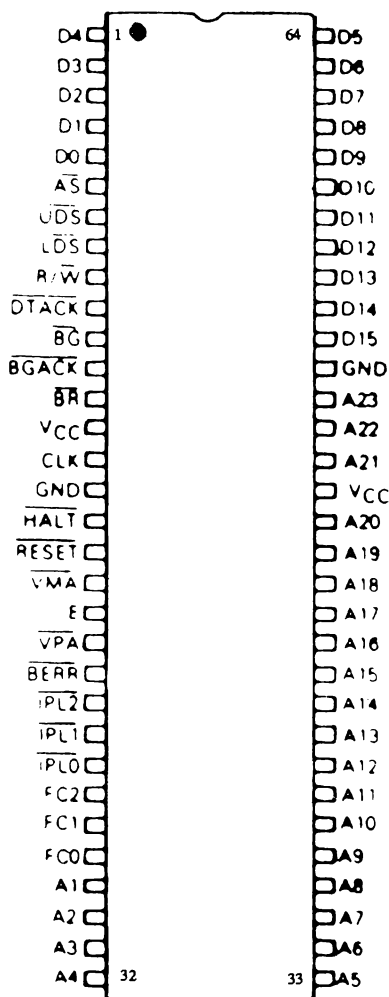



Fig. 5.-Distribuzione dei terminali del microprocessore 68000.

— R/$\overline{\text{W}}$ (Read/Write): mediante tale segnale il 68000 definisce se un ciclo in esecuzione è di lettura o scrittura;

— $\overline{\text{AS}}$ (Addres Strobe): quando esiste un livello basso, che è quello attivo, su questa linea, la CPU comunica all'esterno la presenza di un indirizzo valido sul bus degli indirizzi, per le periferiche asincrone;

— $\overline{\text{DTACK}}$ (Data Transfer Acknowledge): questo segnale di ingresso, informa il 68000 della fine di un trasferimento di dati. Quando la CPU riconosce DTACK, durante un ciclo di lettura, raccoglie l'informazione dal bus dei dati e termina il ciclo. Il riconoscimento, in scrittura, determina la fine del ciclo.

- Bus di controllo sincrono:
  - E (Enable): su questa uscita, la CPU genera un segnale sincrono, con una frequenza di clock fissa. Si tratta di un segnale fondamentale per le periferiche sincrone del 6800;
  - $\overline{\text{VPA}}$ (Valid Peripheral Address): mediante tale segnale di ingresso, si comunica alla CPU che l'elemento indirizzato è una periferica sincrona della famiglia 6800;
  - $\overline{\text{VMA}}$ (Valid Memory Address): segnale di uscita che indica alle periferiche del 6800 che esiste un indirizzo valido sul bus degli indirizzi e che la CPU è sincronizzata per la sua abilitazione. $\overline{\text{VMA}}$ è un segnale di risposta a $\overline{\text{VPA}}$.
- Linee di controllo dello stato del microprocessore: sono tre segnali che escono dalla CPU che forniscono informazioni sulle modalità di lavoro (utente o supervisore), e se la zona a cui si accede corrisponde a programmi o dati. In realtà si controllano quattro mappe di memoria da 16 Mbytes ciascuna. L'informazione fornita da questi terminali, detti FC0, FC1, e FC2 è valida, sempre che il segnale $\overline{\text{AS}}$ si trovi attivato.
- Linee di controllo degli interrupts: questo gruppo di terminali, costituiti da quelli detti IPL0, IPL1, e IPL2, hanno il compito di ricevere il livello di priorità del dispositivo che richiede un interrupt alla CPU.
- Linee di controllo DMA: le linee che comprendo questa funzione, servono per cedere il controllo dei bus del sistema a elementi in comune con altri microprocessori, o per l'accesso diretto alla memoria:
  - $\overline{\text{BR}}$ (Bus request): è un ingresso al 68000 che gli comunica la richiesta da parte di un dispositivo esterno del controllo dei bus del sistema;
  - $\overline{\text{BG}}$ (Bus Grant): è un'uscita con cui la CPU informa di avere ricevuto una richiesta di controllo del bus;
  - $\overline{\text{BGACK}}$: (Bus Grant Acknowledge): la periferica, mediante questo segnale, conferma alla CPU che ha preso il controllo

TABELLA III

SIMBOLO	*DENOMINAZIONE*	*COMPITO*	*STATO ATTIVO*	*INGRES-SO USCI-TA (I o O)*	*TRHEE STATE*
A1-A23	BUS DEGLI INDIRIZZI	INDIRIZZAMENTO	ALTO	O	SI
D0-D15	BUS DEI DATI	TRASFERIMENTO IN- FORMAZIONE	ALTO	I/O	SI
$\overline{AS}$	INDIRIZZO ATTIVATO	SEGNALA PRESENZA DI INDIRIZZO	BASSO	O	SI
R/$\overline{W}$	LETTURA / SCRITTU- RA	INDICA CICLO LET- TURA O SCRITTURA	ALTO / BASSO	O	SI
$\overline{UDS}$ e $\overline{LDS}$	DATO PIU' O MENO SIGNIFICATIVO	SEGNALA BYTE CUI ACCEDERE	BASSO	O	SI
$\overline{DTACK}$	RICONOSCIMENTO TRASFERIMENTO DATI	INDICA TRASFERI- MENTO DI DATI	BASSO	I	NO
$\overline{BR}$	RICHIESTA DI BUS	RICHIEDE I BUS	BASSO	I	NO
$\overline{BG}$	CONCESSIONE DI BUS	CONCEDE I BUS	BASSO	O	NO
$\overline{BGACK}$	RICONOSCIMENTO CONCESSIONE DI BUS	SEGNALA IL POSSES- SO DEI BUS	BASSO	I	NO
$\overline{IPL0}$ $\overline{IPL1}$ $\overline{IPL2}$	LIVELLO PRIORITA- RIO DI INTERRUPT	INDICANO IL LIVEL- LO DI INTERRUPT DEL RICHIEDENTE	BASSO	I	NO
$\overline{BERR}$	ERRORE DI BUS	INDICA UN PROBLE- MA NEL CICLO	BASSO	I	NO
$\overline{RESET}$	RESET	RICHIESTA DI RESET	BASSO	I/O	NO
$\overline{HALT}$	ALT	RICHIESTA DI ALT	BASSO	I/O	NO
E	ABILITAZIONE	SEGNALE DI CLOCK (6800)	ALTO	O	NO
VMA	INDIRIZZO DI MEMO- RIA VALIDO	VALIDAZIONE DEL- L'INDIRIZZO (6800)	BASSO	O	SI
VPA	INDIRIZZO DI PERI- FERICA VALIDO	VALIDAZIONE DEL- LA PERIFERICA (6800)	BASSO	I	NO

SIMBOLO	*DENOMINAZIONE*	*COMPITO*	*STATO ATTIVO*	*INGRES-SO USCI-TA (I o O)*	*TRHEE STATE*
FC0 FC1 FC2	USCITA CODICE DI FUNZIONE	INDICANO MODO E CICLO	ALTO	O	SI
CLK	CLOCK	SEGNALE DI CLOCK DI RIFERIMENTO	ALTO	I	NO
V_{CC} e GND	+5V ±5% e MASSA	ALIMENTAZIONE	—	I	—

Tabella III.-Tabella riassuntiva delle caratteristiche più rappresentative dei 64 terminali del 68000.

dei bus. A partire da tale istante, disattiva $\overline{BR}$ e controlla i bus, mentre mantiene attivo il segnale $\overline{BGACK}$. Quando viene disattivato $\overline{BR}$, il 68000 disattiva $\overline{BG}$.

- Linee di controllo del sistema:
  - — $\overline{RESET}$: è una linea bidirezionale, cioè sia la CPU, sia un dispositivo esterno, è in grado di attivarlo e originare una reinizializzazione del sistema;
  - — $\overline{HALT}$: quando funziona come ingresso ed è attivato dall'esterno, permette di implementare un programma passo a passo, per debuggarlo;
  - — $\overline{BERR}$ (Bus Error): si tratta di un segnale di ingresso che informa il microprocessore del verificarsi di un problema nel sistema, come un accesso illegale alla memoria, o l'errore nella risposta di qualche periferica.

Repertorio delle istruzioni

Il 68000 è stato progettato con un repertorio di sole 56 istruzioni, che sono state ampliate nei modelli più recenti come il 68010. Questo ridotto numero di istruzioni, combinato con l'elevata varietà dei modi di indirizzamento e di dati, offre al programmatore un potente strumento per sviluppare il software dei sistemi basati su tale microprocessore. Le varie istruzioni, suddivise per campo funzionale, si possono raggruppare come segue:

- a) istruzioni aritmetiche: ADD, ADDA, ADDI, ADDQ, ADDX,

MULS, DIVU, DIVS, EXT, CLR, CMP, CMPA, CMPI, CMPM, TST, e TAS;

- b) istruzioni logiche: AND, ANDI, OR, ORI, EOR, EORI, e NOT;
- c) istruzioni di scorrimento e rotazione: ASI, ASR, LSL, LSR, ROL, ROR, ROXL, e ROXR;
- d) istruzioni per la manipolazione di bit: BTST, BSET, BCLR, e BCHG;
- e) istruzioni di trasferimento: MOVE, MOVEM, MOVEP, MOVEQ, EXG, SWAP, LEA, e PEA;
- f) istruzioni per il controllo sequenziale del programma: JMP, BRA, NOP, Bcc, DBcc, e Scc;
- g) istruzioni per il controllo delle subroutines: JSR, BSR, RTS, LINK, e UNLK;
- h) istruzioni per il controllo del sistema: RESET, STOP, RTE, MOVE-SR, ANDI-SR, ORI-SR, EORI-SR, TRAP, TRAPV, e CHK.

Modi di indirizzamento

Il 68000 dispone di 14 diversi modi per l'indirizzamento degli operandi, e risulta, sotto questo aspetto, uno dei più potenti della sua categoria. Tale varietà consente, ai programmatori abili, possibilità straordinarie per ottimizzare le loro realizzazioni software. I modi di indirizzamento si possono suddividere in 6 categorie:

- 1. indirizzamento diretto tramite registro;
- 2. indirizzamento indiretto tramite registro;
- 3. indirizzamento assoluto;
- 4. indirizzamento relativo al contatore di programma;
- 5. indirizzamento immediato;
- 6. indirizzamento implicito.

La famiglia del microprocessore 68000

Come conseguenza dello sviluppo da perte di vari costruttori di periferiche per il microprocessore 68000, nel corso degli ultimi anni, si è avuto un

TABELLA IV

@+: modo di indirizzamento indiretto con postincremento

•: influenzato

—: non influenzato

MNEMONICO	*OPERAZIONE*	*CODICI DI CONDIZIONE*				
		X	*N*	*Z*	*V*	*C*
ABCD	(DESTINAZIONE)$_{10}$+(SORGENTE)$_{10}$ → DESTINAZIONE	•	U	•	U	•
ADD	(DESTINAZIONE)+(SORGENTE) → DESTINAZIONE	•	•	•	•	•
ADDA	(DESTINAZIONE)+(SORGENTE) → DESTINAZIONE	—	—	—	—	—
ADDI	(DESTINAZIONE)+DATO IMMEDIATO → DESTINAZIONE	•	•	•	•	•
ADDQ	(DESTINAZIONE)+DATO IMMEDIATO → DESTINAZIONE	•	•	•	•	•
ADDX	(DESTINAZIONE)+(SORGENTE)+X → DESTINAZIONE	•	•	•	•	•
AND	(DESTINAZIONE) ∧ (SORGENTE) → DESTINAZIONE	—	•	•	0	0
ANDI	(DESTINAZIONE) ∧ DATO IMMEDIATO → DESTINAZIONE	—	•	•	0	0
ASL, ASR	(DESTINAZIONE) SPIAZZATO DA <CONT> → DESTINAZIONE	•	•	•	•	•
Bcc	SE cc ALLORA PC+d → PC	—	—	—	—	—
BCHG	≈(<NUMERO BIT>) DI DESTINAZIONE → Z ≈(<NUMERO BIT>) DI DESTINAZIONE <NUMERO BIT> DI DESTINAZIONE	—	—	•	—	—
BCLR	≈(<NUMERO BIT>) DI DESTINAZIONE → Z 0 → (<NUMERO BIT>) DI DESTINAZIONE	—	—	•	—	—
BRA	PC+d → PC	—	—	—	—	—
BSET	≈(<NUMERO BIT>) DI DESTINAZIONE → Z 1 → (<NUMERO BIT>) DI DESTINAZIONE	—	—	•	—	—
BSR	PC → SP @-; PC+d → PC	—	—	—	—	—
BTST	≈(<NUMERO BIT>) DI DESTINAZIONE → Z	—	—	•	—	—
CHK	SE Dn < 0 O Dn > (<ea>) ALLORA TRAP	—	•	U	U	U
CLR	0 → DESTINAZIONE	—	0	1	0	0
CMP	(DESTINAZIONE)-(SORGENTE)	—	•	•	•	•

MNEMO-NICO	*OPERAZIONE*	*CODICI DI CONDIZIONE*				
		X	*N*	*Z*	*V*	*C*
CMPA	(DESTINAZIONE)-(SORGENTE)	—	•	•	•	•
CMPI	(DESTINAZIONE)-DATO IMMEDIATO	—	•	•	•	•
CMPM	(DESTINAZIONE)-(SORGENTE)	—	•	•	•	•
DBcc	SE ≈cc ALLORA Dn-1 → Dn; SE Dn ≠-1 ALLORA PC+d → PC	—	—	—	—	—
DIVS	(DESTINAZIONE)/(SORGENTE) → DESTINAZIONE	—	•	•	•	0
DIVU	(DESTINAZIONE)/(SORGENTE) → DESTINAZIONE	—	•	•	•	0
EOR	(DESTINAZIONE) ⊕ (SORGENTE) → DESTINAZIONE	—	•	•	0	0
EORI	(DESTINAZIONE) ⊕ DATO IMMEDIATO → DESTINA-ZIONE	—	•	•	0	0
EXG	Rx ↔ Ry	—	—	—	—	—
EXT	(DESTINAZIONE) SEGNO ESTESO → DESTINAZIONE	—	•	•	0	0
JMP	DESTINAZIONE → PC	—	—	—	—	—
JSR	PC → SP @-; DESTINAZIONE → SP	—	—	—	—	—
LEA	DESTINAZIONE → An	—	—	—	—	—
LINK	An → SP @-; SP → An; SP+d → SP	—	—	—	—	—
LSL, LSR	(DESTINAZIONE) SPIAZZATA DA <CONT> → DESTI-NAZIONE	•	•	•	0	•
MOVE	(SORGENTE) → DESTINAZIONE	—	•	•	0	0
MOVE A CCR	(SORGENTE) → CCR	•	•	•	•	•
MOVE A SR	(SORGENTE) → SR	•	•	•	•	•
MOVE ?	SR → DESTINAZIONE	—	—	—	—	—
MOVE USP	USP → An; An → USP	—	—	—	—	—
MOVEA	(SORGENTE) → DESTINAZIONE	—	—	—	—	—
MOVEM	REGISTRI → DESTINAZIONE (SORGENTE) → DESTINAZIONE	—	—	—	—	—
MOVEP	(SORGENTE) → DESTINAZIONE	—	—	—	—	—

MNEMONICO	*OPERAZIONE*	*CODICI DI CONDIZIONE*				
		X	*N*	*Z*	*V*	*C*
MOVEQ	DATO IMMEDIATO → DESTINAZIONE	—	•	•	0	0
MULS	(DESTINAZIONE)•(SORGENTE) → DESTINAZIONE	—	•	•	0	0
MULU	(DESTINAZIONE)•(SORGENTE) → DESTINAZIONE	—	•	•	0	0
MBCD	0-(DESTINAZIONE)$_{10}$-X → DESTINAZIONE	•	U	•	U	•
NEG	0-(DESTINAZIONE) → DESTINAZIONE	•	•	•	•	•
NEGX	0-(DESTINAZIONE)-X → DESTINAZIONE	•	•	•	•	•
NOP	—	—	—	—	—	—
NOT	≈(DESTINAZIONE) → DESTINAZIONE	—	•	•	0	0
OR	(DESTINAZIONE) E (SORGENTE) → DESTINAZIONE	—	•	•	0	0
ORI	(DESTINAZIONE) E DATO IMMEDIATO → DESTINAZIONE	—	•	•	0	0
PEA	(DESTINAZIONE) → SP@-	—	—	—	—	—
RESET	—	—	—	—	—	—
ROL, ROR	(DESTINAZIONE) RUOTATO DA <CONT> → DESTINAZIONE	—	•	•	0	•
ROXL, ROXR	(DESTINAZIONE) RUOTATO DA <CONT> → DESTINAZIONE	•	•	•	0	•
RTE	SP @+ → SR; SP @+ → PC	•	•	•	•	•
RTR	SP @+ → CC; SP @+ → PC	•	•	•	•	•
RTS	SP @+ → PC	—	—	—	—	—
SBCD	(DESTINAZIONE)$_{10}$-(SORGENTE)$_{10}$-X → DESTINAZIONE	•	U	•	U	•
Scc	UN BYTE A 00 O FF IN ACCORDO CON LA CONDIZIONE	—	—	—	—	—
STOP	DATO IMMEDIATO → SR; STOP	•	•	•	•	•
SUB	(DESTINAZIONE)-(SORGENTE) → DESTINAZIONE	•	•	•	•	•
SUBA	(DESTINAZIONE)-(SORGENTE) → DESTINAZIONE	—	—	—	—	—
SUBI	(DESTINAZIONE)-DATO IMMEDIATO → DESTINAZIONE	•	•	•	•	•

MNEMO-NICO	OPERAZIONE	CODICI DI CONDIZIONE				
		X	N	Z	V	C
SUBQ	(DESTINAZIONE)-DATO IMMEDIATO → DESTINAZIONE	•	•	•	•	•
SUBX	(DESTINAZIONE)-(SORGENTE)-X → DESTINAZIONE	•	•	•	•	•
SWAP	REGISTRO [31:16] ↔ REGISTRO [15:0]	—	•	•	0	0
TAS	(DESTINAZIONE) TESTATA → CC; 1 → [7] DI DESTINAZIONE	—	•	•	0	0
TRAP	PC → SSP @; SR → SSP @-; (VETTORE) → PC	—	—	—	—	—
TRAPV	SE V ALLORA TRAP	—	—	—	—	—
TST	(DESTINAZIONE) TESTATA → CC	—	•	•	0	0
UNLK	An → SP; SP @+ → An	—	—	—	—	—

Tabella IV.-Operazioni relative ad ogni istruzione, e funzionamento dei flag di condizione.

veloce potenziamento della sua famiglia. D'altra parte, la sua flessibilità aumenta notevolmente, per la compatibilità degli elementi della famiglia del 6800 (PIA, ACIA, CONTROLLER CRT, ecc.). Attualmente la famiglia del 68000 è costituita dai seguenti circuiti integrati:

- 1. 68000 (CPU a 16 bit);
- 2. 68008 (CPU a 8/16 bit);
- 3. 68010 (CPU con memoria virtuale);
- 4. 68020 (CPU a 32 bit);
- 5. 68200 (microelaboratore monochip);
- 6. 68881 (coprocessore matematico a virgola mobile);
- 7. 68120 e 68121 IPC (controller intelligente di periferiche);
- 8. 68230 MCU, PI/T (interfaccia parallela e timer);
- 9. 68430 DMAI (interfaccia DMA);
- 10. 68440 DDMA (doppio accesso diretto alla memoria);
- 11. 68450 DMAC (controller DMA);
- 12. 68451 MMU (unità di utilizzazione della memoria);
- 13. 68452 BAM (modulo di arbitraggio del bus);

- 14. 68454 IMDC;
- 15. 68459 DPLL;
- 16. 68562 DUSCC;
- 17. 68564 SIO (I/O seriale);
- 18. 68590 LANCE (controller per rete ETHERNET);
- 19. 68652 MPCC (controller di comunicazione con multiprotocollo);
- 20. 68561 MPCC II;
- 21. 68653 PGC;
- 22. 68661 EPCI;
- 23. 68681 DUART (UART a doppio canale);
- 24. 68901 MFP (periferica multifunzione).

Anche se qualcuno di questi componenti non è ancora in produzione, lo sarà in breve tempo.

Il microprocessore 8086

I microprocessori a otto bit sono attualmente i più utilizzati, sia in campo professionale, sia in quello informatico e dell'automazione. I microprocessori a 16 bit sono stati finora utilizzati esclusivamente in campo professionale, per il fatto che, tra l'altro, la realizzazione di una CPU con questi microprocessori è costosa e complessa, principalmente perchè il microprocessore è costituito da vari chips. Con la comparsa dell'8086 nel 1978 si iniziò a prendere in considerazione i microprocessori a 16 bit, poichè tale µP presentava notevoli vantaggi sui predecessori a 16 bit.

Le caratteristiche principali dell'8086 sono:

- il suo bus dei dati è a 16 bit;
- la sua capacità di indirizzamento raggiunge 1 Mbyte (1.048.576 bytes);
- è fabbricato in tecnologia HMOS (MOS ad alta velocità), il che permette di operare ad una frequenza minima di 5 MHz, e consente tempi di esecuzione delle istruzioni più brevi di 500 nsec circa;
- è concepito per lavorare con linguaggi ad alto livello (Basic, Cobol, Fortran, o Pascal), e per l'utilizzo in microelaboratori;
- possiede un vasto repertorio di istruzioni che gli permettono di

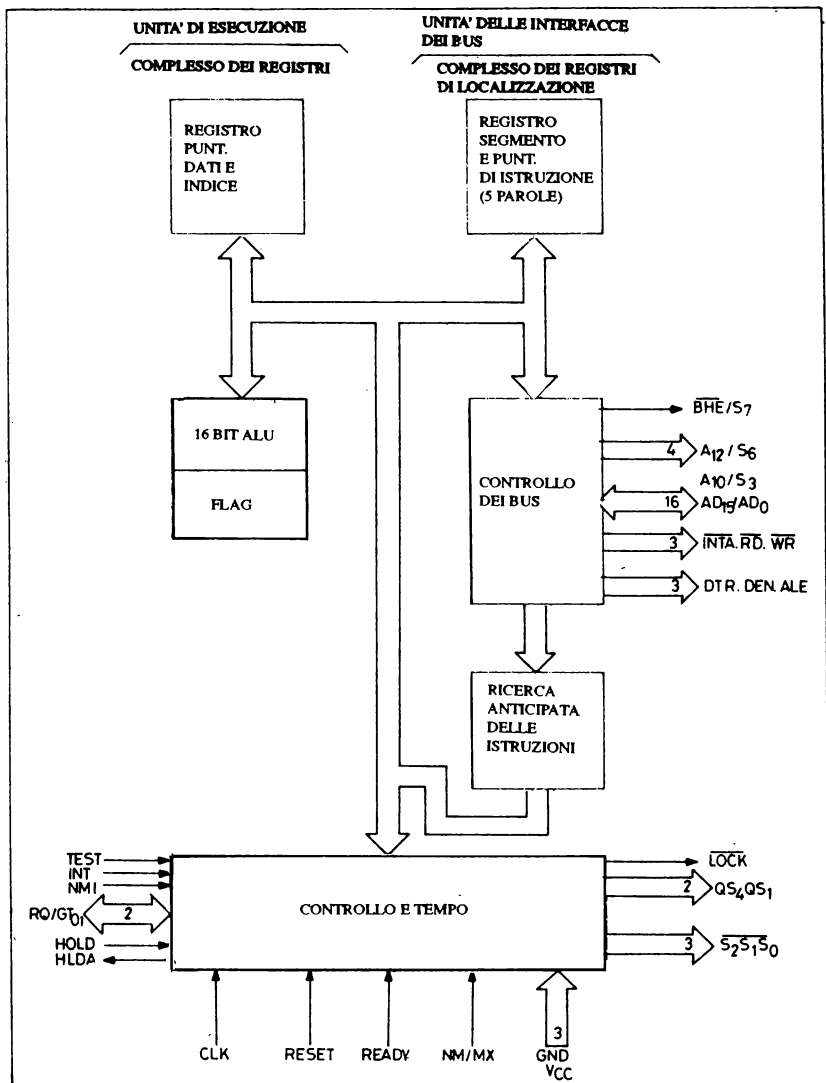



Fig. 6.-Blocchi interni del microprocessore 8086.

operare sia ad 8 che a 16 bit;

- necessita di alimentazione unica a +5 V.

Descrizione dei terminali dell'8086

L'8086 è inserito in un contenitore con il classico formato a 40 piedini, le cui funzioni sono:

- AD0-AD15: sono i terminali del bus degli indirizzi e dei dati, poichè i segnali sui due bus escono in tempi diversi. Esistono quattro tempi di funzionamento T1, T2, T3, e T4; durante T1 su tali terminali è presente l'indirizzo, da T2 a T4 i dati.
- A16-A19: sono quattro terminali su cui esce la parte alta dell'indirizzo per poter accedere al megabyte di memoria; ciò avviene durante T1. Durante T2, T3, e T4 questi terminali sono utilizzati per indicare l'uso dei registri interni (da S3 a S6).
- $\overline{\text{BHE}}$: nel tempo T1 tale terminale, in combinazione con l'uscita AD0 esegue le seguenti funzioni: BHE=0, AD0=0 (bus dei dati a 16 bit), BHE=1, AD0=0 (il byte inferiore è riferito agli indirizzi pari), BHE=0, AD0=1 (il byte superiore del bus dei dati è riferito agli indirizzi dispari), BHE=1, AD0=1 non ha significato.
- RD: uscita per indicare all'esterno il ciclo di lettura (READ).
- READY: ingresso per adattare il µP alle memorie e periferiche lente (ciò è molto utile nell'8086, a causa della sua elevata velocità).
- INTR: ingresso per interrupt mascherabile.
- NMI: ingresso per interrupt non mascherabile.
- TEST: ingresso di prova associato all'istruzione con lo stesso nome (Test).
- RESET: ingresso di inizializzazione del ciclo di lavoro del µP.
- CLX: ingresso del clock esterno (circa 5 MHz).
- V_{CC} e GND: terminali di alimentazione a +5 V e massa, rispettivamente.
- MN/MX: questo terminale fa variare alcuni segnali,in funzione del fatto che il sistema operi con un solo microprocessore (MN/MX=1) o con più microprocessori (MN/MX=0). Per MN/MX=1 i terminali dal 32 al 24 hanno i compiti ordinatamente sottoelencati (vedere Figg. 10 e 11).

- HOLD: ingresso per richieste di acceso al bus provenienti da qualche periferica.
- HOLDA: uscita di riconoscimento della richiesta di accesso, da parte dell'8086, e concessione della stessa.
- WR: uscita indicatrice del ciclo di scrittura (WRITE).
- M/$\overline{\text{IO}}$: tramite questa uscita settata a uno, il µP attiva la memoria, per eseguire qualche istruzione ad essa riferita, o, se è a zero, attiva i circuiti di INPUT/OUTPUT (I/O) per l'esecuzione di qualche istruzione riferita a tali circuiti, a cui si possono collegare periferiche o blocchi di dati fino a 64 Kbytes, che rappresentano

Fig. 7.-Descrizione dei terminali del microprocessore 8086.

l'estensione dell'I/O.

- DT/R: è un'uscita attraverso la quale il µP indica il ciclo di trasmissione o ricezione di dati da circuiti specializzati, che gli si aggiungono in un sistema monoprocessore, ad esempio un 8287.
- DEN: uscita per abilitare i dati in un sistema monoprocessore che utilizza il trasmettitore 8287.
- ALE: questa uscita abilita il latch degli indirizzi.
- INTA: tramite tale uscita il microprocessore indica ad una periferica che ha richiesto un interrupt, che a partire da tale istante lo stesso viene accettato.

Quando il terminale MN/$\overline{\text{MX}}$=0, i terminali numerati dal 32 al 34 hanno i compiti elencati nell'ordine seguente (si ricorda, per una migliore comprensione, di osservare le Figg. 10 e 11).

- $\overline{\text{RQ}}/\overline{\text{GT0}}$: tramite questo ingresso, un altro µP associato può richiedere l'accesso al bus comune, dopo che il µP precedente ha terminato l'istruzione in corso.
- $\overline{\text{RQ}}/\overline{\text{GT1}}$: per mezzo di tale uscita il microprocessore richiede l'accesso al bus condiviso con vari microprocessori in un sistema multiprocessore.
- LOCK: è un'uscita tramite cui il µP indica agli altri presenti nel sistema, che non possono accedere al bus comune mentre questo si trova in stato zero.
- S0-S2: sono tre terminali che decodificano alcuni degli otto possibili stati sottoelencati:
  - — S2=0, S1=0, S0=0 (riconoscimento di interrupt);
  - — S2=0, S1=0, S0=1 (lettura di circuiti I/O);
  - — S2=0, S1=1, S0=0 (scrittura di circuiti I/O);
  - — S2=0, S1=1, S0=1 (pari, halt);
  - — S2=1, S1=0, S0=0 (codice di accesso a istruzione) (Fetch);
  - — S2=1, S1=0, S0=1 (lettura memoria);
  - — S2=1, S1=1, S0=0 (scrittura memoria);
  - — S2=1, S1=1, S0=1 (non opera).
- QS0 e QS1: sono uscite che indicano lo stato del microprocessore nel trattamento delle istruzioni:
  - — QS1=0, QS0=0 (non opera);
  - — QS1=0, QS0=1 (primo byte del codice operativo dalla coda);

8086	Funciones generales		Interrupt Request		Deta Enable
		INTR	Interrupt Request	DEN	Address Latch
		CLK	System Clock	ALE	Enable
		Vcc	+5V	INTA	Interrupt Acknowledge
AD15-ADO	Address/Data Bus	GND	Ground		
—	Address Bus	**Funzioni per un sistema semplice (Terminale MN/$\overline{MX}$ =1)**		**Funzioni per un sistema complesso (Terminale MN/$\overline{MX}$ =0)**	
A 19/S6	Address/Status				
A 16/S3					
BHE/S7	Bush High Enable/ Status				
MN/$\overline{MX}$	Minimum/Maximum Mode Control	HOLD	Hold Request	$\overline{RQ}$/$\overline{GT}$1.0	Request/Grant Bus Access Control
$\overline{RD}$	Read Control	HLDA	Hold Acknowledge		
$\overline{TEST}$	Wait on Test Control	$\overline{WR}$	Write Control	$\overline{LOK}$	Bus Prior Lock Cont
READY	Wait State Control	M/$\overline{IO}$	Memory I/O Control	$\overline{S2\text{-}S0}$	Bus Cicle Status
RESET	System Reset	DT/$\overline{R}$	Data Transmit/		Instruction Queue
NMI	Non-Maskable	DEN	Receive	QS1. QS0	Status

Tabella V.-Terminali del µP 8086, con il significato dei nomi in inglese. Sono raggruppati in tre blocchi.

— QS1=1, QS0=0 (coda vuota);

— QS1=1, QS0=1 (byte successivo dalla coda. Non era il primo).

Registri dell'8086

L'8086 possiede un repertorio di registri, rappresentato in Fig. 8.

Il tipo di istruzioni di questo µP permette di operare con 8 o 16 bit, in base all'istruzione, per cui il primo blocco di registri costituisce degli accumulatori a 8 bit, detti AH, AL, BH, BL, CH, CL, DH, e DL (A, B, C, D indica l'ordine del registro, H la parte alta e L la parte bassa). Quando questi accumulatori sono trattati come 16 bit vengono definiti AX, BX, CX, DX.

Il blocco di registri successivo contiene il puntatore di stack e altri tre puntatori detti base, sorgente e destinazione, per il movimento dei dati interno, o rispetto alla memoria.

Il terzo blocco contiene il puntatore dell'istruzione che funziona da contatore di programma, e il registro dei flags, che è effettivamente il registro di stato, tipico di tutti i µP. Questi bit indicano il segno del dato elaborato, il carry, l'overflow, la parità, lo stato degli interrupts mascherabili mediante il bit 1 (num. 9), l'autoincremento o decremento di un registro, un bit ausiliario detto trap.

L'ultimo blocco comprende quattro registri a 16 bit, che definiscono in modo indipendente quattro aree di memoria da 64 Kbytes ciascuna.

L'8086 è, come detto, un microprocessore completo e complesso, per cui

AX	AH	AL	ACCUMULATORE	COMPLESSO DEI REGISTRI GENERALI
BX	BH	BL	BASE	
CX	CH	CL	CONTATORE	
DX	DH	DL	DATI	
	SP		STACK POINTER	
	BP		PUNT. DI BASE	
	SI		SCORR. INDICE	
	DI		DEST. INDICE	
	IP		PUNT. DELLA ISTRUZIONE	
	FLAGS	FLAGS		
	CS		COD. DEL SEG.	COMPLESSO DEI REGISTRI PUNT. DEI SEGMENTI
	DS		DATO SEG.	
	SS		STACK SEG.	
	ES		EXTRA SEG.	

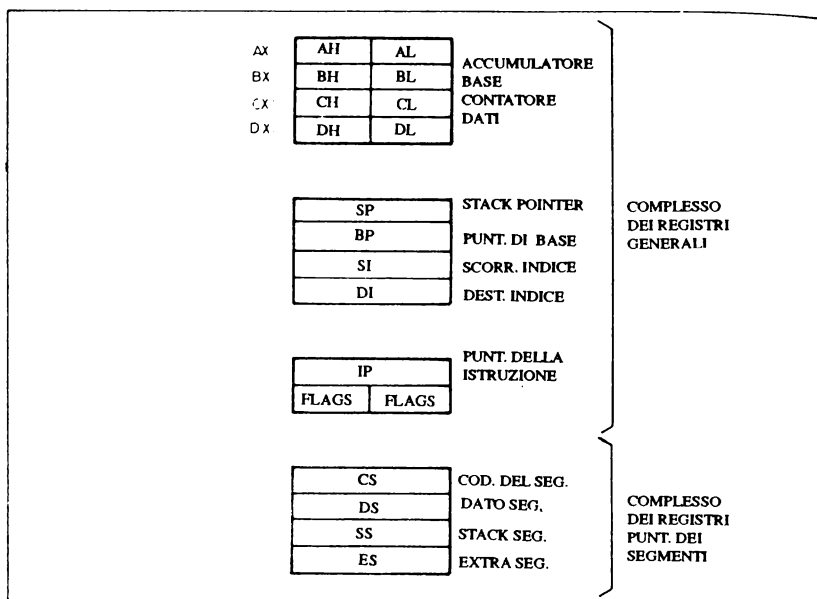



Fig. 8.-Mappa dei registri del microprocessore 8086.

si fornisce solo una panoramica dello stesso, per ampliare la comprensione dei microprocessori. Si rimanda perciò, per chi fosse interessato a questo microprocessore, alle informazioni fornite dai costruttori.

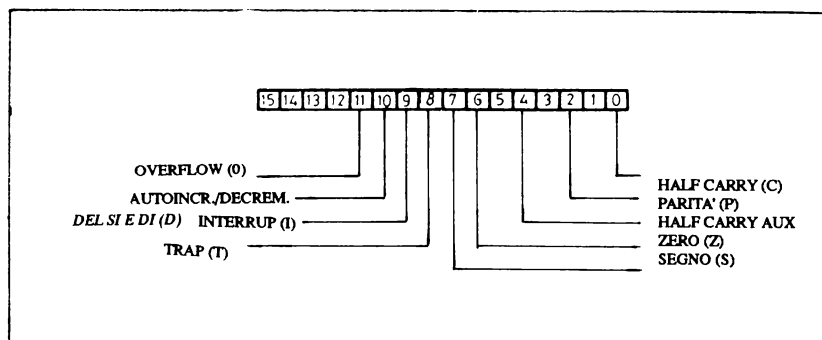



Fig. 9.-Registro di stato del μP 8086.

Il µP 8086 è concepito per essere utilizzato in due modi di configurazione: secondo il sistema mono e multiprocessore.

Nel sistema monoprocessore il µP lavora indipendentemente con la sua circuiteria tipica; è il sistema di lavoro di qualsiasi altro microprocessore. La sua configurazione schematica è rappresentata in Fig. 10.

Tutti i microprocessori della serie 80 hanno molto in comune, ad esempio il circuito 8284 è il tipo di clock utilizzato per gli altri.

Come già detto nella descrizione, il suo bus a 16 bit è comune ai dati e agli indirizzi, in tempi diversi, per cui è necessario un latch che memorizzi l'indirizzo che esce dal bus nel primo ciclo, per poi operare sui dati che escono sul bus, diretti a tale indirizzo.

In realtà, tale sistema è seguito anche da altri microprocessori a 16 bit inseriti in contenitori a 40 terminali. Il circuito integrato 8286 è fondamentalmente un amplificatore bidirezionale del bus dei dati, che opera sul flusso in ingresso e in uscita.

Il sistema di collegamento multiprocessore, è concepito per operare in sistemi molto più complessi e per l'utilizzo con altri microprocessori 8086, in modo interlacciato.

In un sistema multiprocessore, in cui operano indipendentemente vari microprocessori che condividono a volte il bus, esiste sempre uno di essi il cui software è strutturato in modo che funzioni come principale, mentre gli altri gli sono subordinati. In tale sistema, inoltre, è possibile aggiungere ai circuiti

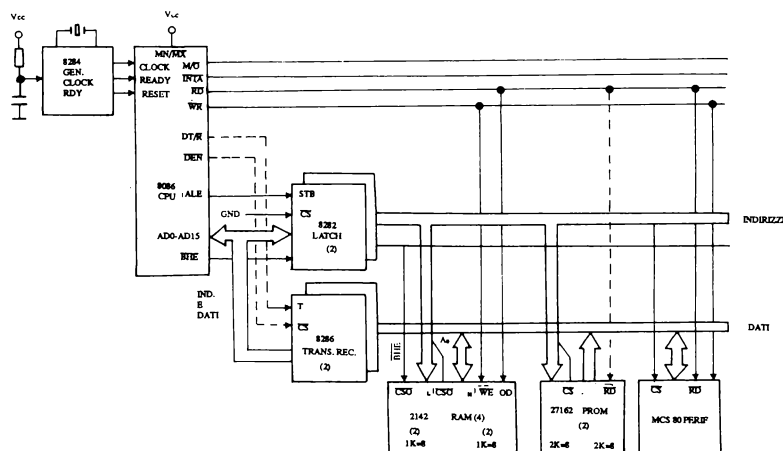



Fig. 10.-Schema a blocchi dei collegamenti dei circuiti di un 8086, secondo il sistema a monoprocessore.

generatori di clock, ai latch e amplificatori del bus, altri circuiti come l'8288 controllore dei bus e l'8285 controllore degli interrupts, che nel µP 8086 sono simili a quelli del predecessore 8085.

Negli schemi delle varie figure sono rappresentati un sistema semplice ed uno più complesso, ed i vari blocchi con tutti i circuiti integrati LSI, da inserire opzionalmente in sistemi basati sul µP 8086, per cui sono stati progettati.

La tendenza di questo microprocessore è quella valida attualmente per tutti i microprocessori a 16 bit, che apre nuove prospettive per il futuro.

Questa tendenza è orientata a fornire una sempre maggior potenza delle istruzioni, per ridurre al massimo il costo del software, e creare gli LSI necessari che, agendo in combinazione con il microprocessore, facilitino il suo collegamento e le sue possibilità di impiego.

Tabella delle istruzioni dell'8086

- Istruzioni di trasferimento:

MOV: trasferisce una parola (16 bit) o un byte (8 bit);
XCHG: scambia una parola o un byte;
LEA: carica un indirizzo in un registro;
LDS: carica un puntatore tramite DS;
LES: carica un puntatore mediante ES;
LAHF: carica i flags;

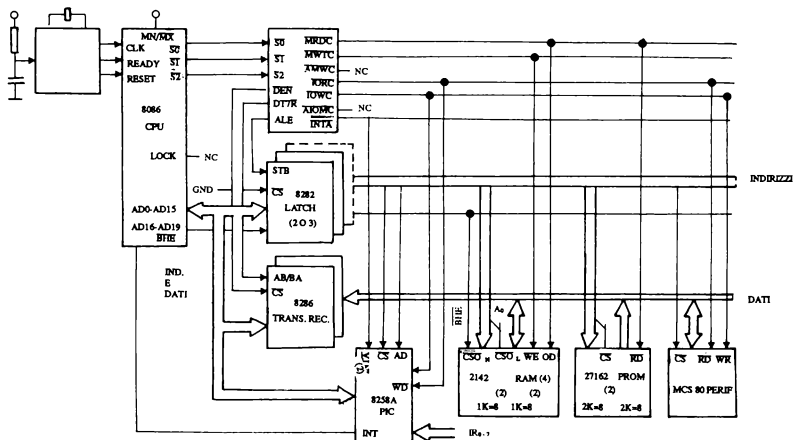



Fig. 11.-Schema a blocchi dei collegamenti dei circuiti di un 8086, in un sistema a multiprocessore.

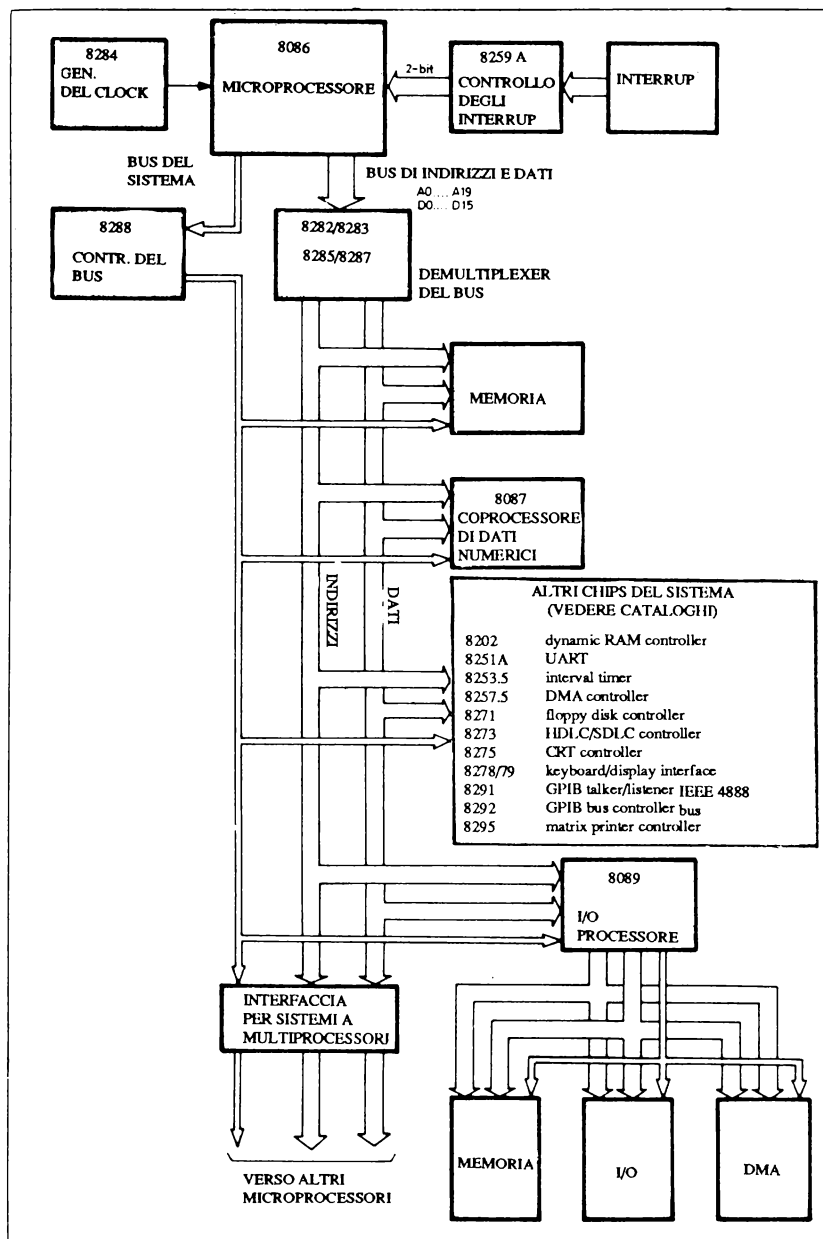



Fig. 12.-Schema dei diversi circuiti integrati in LSI disponibili per il μP 8086.

SAHF: trasferisce i flags;
PUSHF: mette i flags nello stack;
POPF: estrae i flags dallo stack;
XLAT: trasforma un carattere di un codice (EBCDID) in un altro (ASCII).

- *Istruzioni aritmetiche:*
ADD: somma due parole o bytes con o senza segno;
ADC: somma tenendo conto del carry;
SUB: sottrae;
SUBB: sottrae tenendo conto del prestito;
MUL: moltiplicazione di parole o bytes, con o senza segno;
IMUL: moltiplicazione di dati interi;
DIV: divisione di dati a 32 e 16 bit per dati a 16 e 8 bit senza segno;
IDIV: divisione intera con segno;
CMP: confronto di parole o dati;
CBN: estensione di segno passando da 8 a 16 bit;
CWD: estensione di segno passando da 16 a 32 bit;
DAA; regolazione decimale per sommare;
DAS: regolazione decimale per sottrarre;
DEG: negazione di un numero;
AAM: regolazione decimale per moltiplicare;
AAD: regolazione decimale per dividere;
INC: incremento;
DEC: decremento.

- *Istruzioni logiche:*
NOT: negazione;
OR: operazione OR;
AND: operazione AND;
XOR: operazione OR esclusivo;
TEST: operazione AND per verificare i flags.

- *Istruzioni di salto e interrupt:*
JMP: salto incondizionato;
J: salto dipendente da 30 condizioni diverse;
JCX: salto se CX=0;
LOOP: regola di cinque modi alla fine di un loop;
INT: interrupt via software;
INTO interrupt condizionato (se O=Φ overflow);
IRET: ritorno da interrupt;
LODS: trasferisce un dato da AL e AX;
STOS: trasferisce un dato dai registri accumulatori AL e AX;
SCAS: trasferisce un dato con i registri AL e AX;

CMPS: confronta due caratteri;

MOVSB e MOVSM: trasferisce un byte o una parola.

- *Istruzioni di controllo:*

NOP: non opera;

WAIT: per il processore fino a che TEST sia 0;

HALT: per il processore;

LOCK: permette a LOCK di passare a 0 durante l'esecuzione dell'istruzione successiva;

CLD e CLI: settano a 0 i flags D e I, rispettivamente;

STD e STI: settano a 1 i flags D e I, rispettivamente;

CLC e STC: settano a 0 e 1 il flag di carry;

CMC: complementa il flag di carry;

IN: questa istruzione di ingresso trasferisce il dato da una periferica ai registri AL o AX;

OUT: questa istruzione di uscita trasferisce il dato dai registri AL o AX a una periferica;

ROL: rotazione a sinistra;

ROR: rotazione a destra;

RCL: rotazione a sinistra includendo il bit (o flag) di carry del registro di stato;

RCR: rotazione a destra con il bit (o flag) di carry del registro di stato;

CALL: chiamata a subroutine;

RET: ritorno da subroutine;

PUSH: carica un dato nello stack;

POP: estrae un dato dallo stack;

REP: ripetere;

REPE: ripetere se uguale;

REPNE: ripetere se diverso;

REPZ, REPNZ: ripetere se è zero o se non è zero, rispettivamente. Sono istruzioni che elaborano catene di caratteri, utilizzando come puntatori i registri SI e DI.

Significato delle abbreviazioni

Nell'insieme delle istruzioni si utilizzano abbreviazioni e note il cui significato è il seguente:

AL=registro accumulatore a 8 bit;

AX=registro accumulatore a 16 bit;

CX=registro contatore;

DS=segmento dei dati;

ES=segmento extra;
ABOVE=al di sopra (riferito al valore assegnato);
BELOW=al di sotto (riferito al valore assegnato);
GREATER=valore più positivo;
LESS=valore meno positivo (più negativo), riferito al segno dei valori. Se d=1, allora il valore è più negativo di quello del registro. Se d=0, il valore è più positivo di quello del registro. Se W=1, allora l'istruzione è di parola (tiene conto del segno). Se W=0, l'istruzione è di byte.

Condizioni

- se Mod.=11, allora r/m è trattato come un registro;
- se Mod.=00, allora DISP=0 (senza spostamento, eccetto se r/m=110, allora EA indica lo spostamento);
- se Mod.=01, allora (DISP) lo spostamento basso è di 16 bit con segno (non c'è spostamento alto);
- se Mod.=10, allora DESPL ALTO=DESPL BASSO;
- se r/m=000, allora EA=(BX)+(SI)+spostamento;
- se r/m=001, allora EA=(BX)+(DI)+spostamento;
- se r/m=010, allora EA=(BP)+(SI)+spostamento;
- se r/m=011, allora EA=(BP)+(DI)+spostamento;
- se r/m=100, allora EA=(SI)+spostamento;
- se r/m=101, allora EA=(DI)+spostamento;
- se r/m=110, allora EA=(BP)+spostamento;
- se r/m=111, allora EA=(BX)+spostamento;
- se W=01, allora prende il dato come 16 bit;
- se W=11, allora il dato di un byte viene esteso a 16 bit:
- se V=0, allora *Count*=1, se V=1 allora *Count* in CL;
- X=qualsiasi valore;
- prefisso di segmento=001 reg 1101;
- Reg viene assegnato in base alla tabella seguente:

Le istruzioni che si riferiscono ai bit o flags dello status usano 16 bit.
Il simbolo dei flags è:
FLAGS=XXXX(OF): (DF): (IF): (TF): (SF): (ZF): X: (AF): X: (PF): X(CF).

ISTRUZIONI DI TRASFERIMENTO DATI

Istruzione	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
MOV=Move				
Register/memory to/trom register	1 0 0 0 1 0 d w	mod reg r/m		
Inmediate to register/memory	1 1 0 0 0 1 1 w	mod 000 r/m	data	data if w 1
Inmediate to register	1 0 1 1 w reg	data	data if w1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-iow	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		
PUSH=Push:				
Register/memory	1 1 1 1 1 1 1 1	mod 110 r/m		
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			
POP=Pop:				
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			
XCHG=Exchange:				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m		
Register with accumulator	1 0 0 1 0 reg			
IN=Input from:				
Fixed port	1 1 1 0 0 1 0 w	port		
Variable port	1 1 1 0 1 1 0 w			
OUT=Output to:				
Fixed port	1 1 1 0 0 1 1 w	port		
Variable port	1 1 1 0 1 1 1 w			
XLAT=Transiate byte to AL	1 1 0 1 0 1 1 1			
LEA=Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m		
LDS=Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		
LES=Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		
LANF=Load AH with flags	1 0 0 1 1 1 1 1			
SANF=Store AH into flags	1 0 0 1 1 1 1 0			
PUSNF=Push flags	1 0 0 1 1 1 0 0			
POPF=Pop flags	1 0 0 1 1 1 0 1			
DEC=Decrement:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG=Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		
CMP=Compare:				
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Inmediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s w 01
Inmediate with accumulator	0 0 1 1 1 1 0 w	data	data if w 1	
AAS=ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS=Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL=Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL=Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM=ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV=Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV=Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD=ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW=Convert byte to word	1 0 0 1 1 0 0 0			
CWD=Convert word to double word	1 0 0 1 1 0 0 1			

ISTRUZIONI LOGICHE

NOT=Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m
SHL/SAL=Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m
SHR=Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m
SAR=Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m
ROL=Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m
ROR=Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m
RCL=Rotte through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m
RCR=Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m

AND=And:

Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Inmediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
Inmediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	

ISTRUZIONI ARITMETICHE

ADD=Add:

Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Inmediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s w=01
Inmediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1	

ADC=Add with carry:

Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Inmediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s w=01
Inmediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	

INC=Increment:

Register/memory	1 1 1 1 1 1 1 w	mod 000 r/m
Register	0 1 0 0 0 reg	
AAA=ASCII adjust for add	0 0 1 1 0 1 1 1	
DAA=Decimal adjust for add	0 0 1 0 0 1 1 1	

SUB=Subtract:

Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Inmediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s w=01
Inmediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	

SBB=Subtract with borrow

Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Inmediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s w=01
Inmediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

TEST=And function to flags, no result:

Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Inmediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Inmediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	

OR=Or:

Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Inmediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Inmediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	

XOR=Exclusive or:

Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Inmediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Inmediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	

ISTRUZIONI DI CONTROLLO

REP=Repeat	1 1 1 1 0 0 1 z
MOVS=Move byte/word	1 0 1 0 0 1 0 w
CMPS=Compare byte/word	1 0 1 0 0 1 1 w
SCAS=Scan byte/word	1 0 1 0 1 1 1 w
LODS=Load byte/wd to AL/AX	1 0 1 0 1 1 0 w
STDS=Stor byte/wd from AL/A	1 0 1 0 1 0 1 w

ISTRUZIONI DI CONTROLLO DEL TRASFERIMENTO

CALL=Call:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high
Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	
Direct intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	
JMP=Unconditional Jump:			
Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within segment-short	1 1 1 0 1 0 1 1	disp	
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET=Return from CALL:			
Within segment	1 1 0 0 0 0 1 1		
Within seg adding immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment adding inmediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ=Jump on equal/zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE=Jump on less/not greater on equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG=Jump on less or equal/not greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE=Jump on below/not above or equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA=Jump on below or equal/not above	0 1 1 1 0 1 1 0	disp	
JP/JPE=Jump on parity/parity even	0 1 1 1 1 0 1 0	disp	
JO=Jump on overflow	0 1 1 1 0 0 0 0	disp	
JS=Jump on sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ=Jump on not equal/not zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE=Jump on not less/greater or equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG=Jump on not less or equal/greater	0 1 1 1 1 1 1 1	disp	
JNB/JAE=Jump on not below/above or equal	0 1 1 1 0 0 1 1	disp	
JNBE/JA=Jump on not below or equal/above	0 1 1 1 0 1 1 1	disp	
JNP/JPO=Jump on not par/par odd	0 1 1 1 1 0 1 1	disp	
JNO=Jump on not overflow	0 1 1 1 0 0 0 1	disp	
JNS=Jump on not sign	0 1 1 1 1 0 0 1	disp	
LOOP=Loop. CX times	1 1 1 0 0 0 1 0	disp	
LOOPZ/LOOPE=Loop with zero/equal	1 1 1 0 0 0 0 1	disp	

Istruzione	Codice	
LOOPNZ/LOOPNE=Loop white not zero/equal	1 1 1 0 0 0 0 0	disp
JCXZ=Jump on CX zero	1 1 1 0 0 0 1 1	disp
INT=Interrupt		
Type specified	1 1 0 0 1 1 0 1	type
Type 3	1 1 0 0 1 1 0 0	
INTO=Interrupt on overflow	1 1 0 0 1 1 1 0	
IRET=Interrupt return	1 1 0 0 1 1 1 1	
ISTRUZIONI DI CONTROLLO DEL MICROPROCESSORE		
CLC=Clear carry	1 1 1 1 1 0 0 0	
CMC=Complement carry	1 1 1 1 0 1 0 1	
STC=Set carry	1 1 1 1 1 0 0 1	
CLD=Clear direction	1 1 1 1 1 1 0 0	
STD=Set direction	1 1 1 1 1 1 0 1	
CLI=Clear interrupt	1 1 1 1 1 0 1 0	
STI=Set interrupt	1 1 1 1 1 0 1 1	
HLT=Halt	1 1 1 1 0 1 0 0	
WAIT=Wait	1 0 0 1 1 0 1 1	
ESC=Escape (to external device)	1 1 0 1 1 x x x	mod x x x r/m
LOCK=Bus lock prefix	1 1 1 1 0 0 0 0	

Tabella VI.-Istruzioni del µP 8086.

Modi di indirizzamento

I modi fondamentali di indirizzamento dell'8086 sono:

- mediante registro base: l'indirizzo si ottiene mediante la somma del contenuto di un registro base e il dato dell'istruzione;
- immediato: indicato dalla stessa istruzione;
- mediante registro indiretto: l'indirizzo si trova in BX, BP, SI, DI;
- diretto: l'indirizzo è nella stessa istruzione;
- relativo: utilizzato per istruzioni che necessitano salti, e per l'accesso a subroutine;
- mediante registro indicizzato: l'indirizzo si ottiene sommando a un registro indice il dato dell'istruzione;
- mediante registro base indicizzato: l'indirizzo finale è la somma del contenuto del registro base, dell'indice, e del dato dell'istruzione.

CAPITOLO 8

CONCLUSIONI

I microprocessori, che sono realmente unità centrali di elaborazione (CPU) realizzati con uno (o pochi) circuiti integrati, sono componenti la cui evoluzione è molto legata ai progressi della tecnologia di fabbricazione dei circuiti integrati e all'architettura degli elaboratori.

Il microprocessore è un componente elettronico, un circuito integrato, che esegue le funzioni di unità centrale di un elaboratore. Quando al µP si aggiungono circuiti di memoria, e di ingresso/uscita si ottiene un µC, ossia un elaboratore realizzato con un microprocessore.

Il microcomputer è il livello inferiore nella classificazione degli elaboratori per la sua dimensione (e prezzo): microcomputer, minicomputer, e grande elaboratore (o mainframe, come viene normalmente chiamato). L'obiettivo della ricerca nel campo degli elaboratori è di ottenere maggiori velocità di elaborazione e potenza di calcolo, utilizzando a questo scopo componenti elettronici con tempo di propagazione minore, minori dimensioni, e minore consumo, e organizzando la macchina con architetture più efficienti.

Nello stesso tempo i progettisti di minielaboratori, basandosi sull'architettura di quelli grandi, realizzano semplificazioni degli stessi e utilizzano altri componenti e altre tecnologie che, anche se diminuiscono le prestazioni complessive dell'apparecchio, portano a notevoli riduzioni delle dimensioni e del prezzo.

Infine, i progettisti di microelaboratori semplificano al massimo questi processi per cui tutto il circuito può essere realizzato con un solo circuito integrato.

I progressi nella tecnologia di fabbricazione dei circuiti integrati, che permettono di conglobare in un solo chip una gran quantità di circuiti attivi, con-

sente a un microprocessore di effettuare prestazioni che superano quelle dei minielaboratori di qualche anno fa. Lo stesso fatto accade per i minielaboratori rispetto ai mainframes.

Glossario

La forma con cui i temi sono stati esposti è stata seria e chiara, senza essere pedante, con un approfondimento progressivo per evitare paure infondate. Con la stessa finalità si inserisce in quest'ultimo capitolo un glossario, concepito per agevolare la comprensione dei termini meno conosciuti, cui riferirsi quando nel testo compare qualche termine nuovo.

ACKNOWLEDGE: risposta affermativa. Segnale di controllo utilizzato per completare una sequenza di conferma di uno scambio di indicazioni e segnali di controllo.

ADDRESS: indirizzo. Numero che indica la posizione di una parola in memoria.

AND: operazione logica definita dalla regola seguente: se A e B sono 1, C è 1; altrimenti è 0.

BISTABILE: dispositivo con due possibili stati che possono essere invertiti mediante un segnale esterno.

BIT: contrazione di binary digit (numero binario). Unità di informazione che può assumere due valori o stati distinti.

BYTE: ottetto. Termine che rappresenta una parte misurabile di numeri binari consecutivi.

CARRY: riporto delle varie operazioni aritmetiche che può effettuare un microprocessore.

CHIP: circuito integrato.

CLOCK: generatore di impulsi. Dispositivo che genera una base dei tempi utilizzata per fornire gli impulsi sequenziali fondamentali per le operazioni di un elaboratore, il cui modo di lavorare è sequenziale.

CPU: unità centrale di processo. Modulo che ha il compito di cercare, decodificare ed eseguire istruzioni. Comprende un'unità di controllo, una unità aritmetica e logica (ALU) e dispositivi affini (registri, clock, ecc.).

DMA: direct memory access. Accesso diretto alla memoria.

FIRMWARE: supporto logico inalterabile. Programma esistente in ROM.

HARDWARE: componenti, dispositivi fisici. Apparecchiatura fisica contrapposta ai programmi dell'elaboratore o ai metodi per il suo utilizzo.

HMOS: i progressi della tecnologia MOS hanno permesso di aumenta-

re la densità di integrazione della NMOS, e di quadruplicare il suo rapporto velocità/consumo.

LATCH: dispositivo fisico che cattura l'informazione e la trattiene.

KILOBYTE (K): 1024 bytes.

LSI (Large Scale Integration): integrazione a grande scala.

MEGABYTE (M): 1048576 bytes.

µP:microprocessore. E' fondamentalmente un circuito integrato che si può utilizzare come circuito logico programmabile, o come unità centrale di processo di un microelaboratore.

NAND: porta NON-AND. Operazione logica definita dalla regola seguente: se tutti gli ingressi sono 1, l'uscita è 0; altrimenti è 1.

NOT: funzione negazione, per cui tutti i bit o ingressi sono negati.

OR: porta OR. Elemento logico in cui la variabile presente nell'unico segnale binario di uscita è la disgiunzione delle variabili presenti nei due o più segnali binari di ingresso; cioè l'uscita sarà 1 quando almeno un ingresso è 1.

PIN: terminale di collegamento all'esterno dei circuiti integrati.

READY: Segnale utilizzato come ingresso o uscita nei microprocessori, per comunicare o ricevere la conferma che lo stesso o una periferica è pronto.

RESET: azzeramento. In programmazione, azione di mettere a zero un contatore o portare un indicatore a qualche condizione stabile.

SILICIO: elemento molto comune utilizzato come base per la costruzione dei circuiti integrati.

SOFTWARE: supporto logico, insieme di programmi. Complesso di programmi e procedimenti che si inseriscono in un apparecchio di elaborazione dati, e che ne rendono possibile l'utilizzazione.

STACK: catasta. Struttura LIFO che memorizza l'informazione in ordine cronologico.

VLSI (Very Large Scale Integration): Integrazione a scala molto grande.

CONTIENE LA GUIDA A:
SIMBOLI, UNITÀ DI MISURA,
TABELLE DI CONVERSIONE
E CODICI COLORE
DEI COMPONENTI ELETTRONICI

TABELLE
E
CODICI

GUIDA AI SIMBOLI
UNITÀ DI MISURA
TABELLE DI CONVERSIONE
E CODICI COLORE
DEI COMPONENTI ELETTRONICI

La parte hardware che costituisce il cuore effettivo di qualsiasi apparecchiatura di elaborazione è la cosiddetta CPU. Tale dispositivo, a sua volta, è gestito da un particolare componente integrato noto come MPU o, più generalmente, come microprocessore.
I microprocessori altro non sono che dei chip integrati su larga scala dotati ognuno di una sua specificità costruttiva. Pur se diversi strutturalmente, i vari microprocessori si rifanno tutti ad un'unica logica funzionale che viene ampiamente descritta nel presente volume. Sono inoltre descritte le caratteristiche principali e la tecnologia di fabbricazione di vari tipi di microprocessori, sia a 8 che a 16 bit. Ciò, di primo acchito, permette di confrontare tra loro i vari componenti, consentendo così un'utile verifica atta a facilitarne l'utilizzo e la scelta, nel caso in cui si desideri effettuare qualche esperimento basato su di essi.

GRUPPO EDITORIALE JACKSON

ISBN 88-256-0068-2

9 788825 600681

L. 12.500 Cod. 072E

EH17

MICROPROCESSORI